



man pages section 3: Curses Library Functions

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 817-0663-10
April 2003

Copyright 2003 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, docs.sun.com, AnswerBook, AnswerBook2, and Solaris are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

Federal Acquisitions: Commercial Software—Government Users Subject to Standard License Terms and Conditions.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 2003 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. Tous droits réservés

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées du système Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, docs.sun.com, AnswerBook, AnswerBook2, et Solaris sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REpondre A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.



030212@5115



Contents

Preface 29

Curses Library Functions 35

addch(3CURSES) 36
addch(3XCURSES) 39
addchnstr(3CURSES) 41
addchnstr(3XCURSES) 42
addchstr(3CURSES) 44
addchstr(3XCURSES) 45
addnstr(3CURSES) 47
addnstr(3XCURSES) 48
addnwstr(3CURSES) 50
addnwstr(3XCURSES) 51
addstr(3CURSES) 53
addstr(3XCURSES) 54
addwch(3CURSES) 56
add_wch(3XCURSES) 59
addwchnstr(3CURSES) 61
add_wchnstr(3XCURSES) 63
addwchstr(3CURSES) 65
add_wchstr(3XCURSES) 67
addwstr(3CURSES) 69
addwstr(3XCURSES) 70
adjcurspos(3CURSES) 72
arc(3PLOT) 73
attr_get(3XCURSES) 76

attroff(3CURSES) 78
attr_off(3XCURSES) 80
attroff(3XCURSES) 82
attron(3CURSES) 84
attr_on(3XCURSES) 86
attron(3XCURSES) 88
attrset(3CURSES) 90
attr_set(3XCURSES) 92
attrset(3XCURSES) 94
baudrate(3CURSES) 96
baudrate(3XCURSES) 98
beep(3CURSES) 99
beep(3XCURSES) 100
bkgd(3CURSES) 101
bkgd(3XCURSES) 102
bkgdset(3CURSES) 104
bkgdset(3XCURSES) 105
bkgrnd(3XCURSES) 107
bkgrndset(3XCURSES) 109
border(3CURSES) 111
border(3XCURSES) 113
border_set(3XCURSES) 115
bottom_panel(3CURSES) 118
box(3CURSES) 119
box(3PLOT) 121
box(3XCURSES) 124
box_set(3XCURSES) 126
can_change_color(3CURSES) 129
can_change_color(3XCURSES) 132
cbreak(3CURSES) 135
cbreak(3XCURSES) 138
chgat(3XCURSES) 139
circle(3PLOT) 141
clear(3CURSES) 144
clear(3XCURSES) 145
clearok(3CURSES) 146
clearok(3XCURSES) 149
closepl(3PLOT) 151

| | |
|-------------------------|-----|
| closevt(3PLOT) | 154 |
| clrtobot(3CURSES) | 157 |
| clrtobot(3XCURSES) | 158 |
| clrtoeol(3CURSES) | 159 |
| clrtoeol(3XCURSES) | 160 |
| color_content(3CURSES) | 161 |
| color_content(3XCURSES) | 164 |
| COLOR_PAIR(3XCURSES) | 167 |
| COLOR_PAIRS(3XCURSES) | 170 |
| COLORS(3XCURSES) | 173 |
| color_set(3XCURSES) | 176 |
| COLS(3XCURSES) | 178 |
| cont(3PLOT) | 179 |
| copywin(3CURSES) | 182 |
| copywin(3XCURSES) | 183 |
| current_field(3CURSES) | 185 |
| current_item(3CURSES) | 187 |
| curs_addch(3CURSES) | 189 |
| curs_addchstr(3CURSES) | 192 |
| curs_addstr(3CURSES) | 193 |
| curs_addwch(3CURSES) | 194 |
| curs_addwchstr(3CURSES) | 197 |
| curs_addwstr(3CURSES) | 199 |
| curs_alecompat(3CURSES) | 200 |
| curs_attr(3CURSES) | 201 |
| curs_beep(3CURSES) | 203 |
| curs_bkgd(3CURSES) | 204 |
| curs_border(3CURSES) | 205 |
| curs_clear(3CURSES) | 207 |
| curs_color(3CURSES) | 208 |
| curscr(3XCURSES) | 211 |
| curs_delch(3CURSES) | 212 |
| curs_deleteln(3CURSES) | 213 |
| curses(3CURSES) | 214 |
| curses(3XCURSES) | 229 |
| curs_getch(3CURSES) | 240 |
| curs_getstr(3CURSES) | 245 |
| curs_getwch(3CURSES) | 246 |

curs_getwstr(3CURSES) 251
curs_getyx(3CURSES) 252
curs_inch(3CURSES) 253
curs_inchstr(3CURSES) 254
curs_initscr(3CURSES) 255
curs_inopts(3CURSES) 257
curs_insch(3CURSES) 260
curs_insstr(3CURSES) 261
curs_instr(3CURSES) 263
curs_inswch(3CURSES) 264
curs_inswstr(3CURSES) 265
curs_inwch(3CURSES) 267
curs_inwchstr(3CURSES) 268
curs_inwstr(3CURSES) 269
curs_kernel(3CURSES) 270
curs_move(3CURSES) 272
curs_outopts(3CURSES) 273
curs_overlay(3CURSES) 276
curs_pad(3CURSES) 277
curs_printw(3CURSES) 279
curs_refresh(3CURSES) 280
curs_scanw(3CURSES) 282
curs_scr_dump(3CURSES) 283
curs_scroll(3CURSES) 285
curs_set(3CURSES) 286
curs_set(3XCURSES) 288
curs_slk(3CURSES) 289
curs_termattrs(3CURSES) 291
curs_termcap(3CURSES) 293
curs_terminfo(3CURSES) 295
curs_touch(3CURSES) 299
curs_util(3CURSES) 301
curs_window(3CURSES) 303
cur_term(3XCURSES) 306
data_ahead(3CURSES) 307
data_behind(3CURSES) 308
def_prog_mode(3CURSES) 309
def_prog_mode(3XCURSES) 311

| | |
|-----------------------------|-----|
| def_shell_mode(3CURSES) | 312 |
| def_shell_mode(3XCURSES) | 314 |
| delay_output(3CURSES) | 315 |
| delay_output(3XCURSES) | 317 |
| delch(3CURSES) | 318 |
| delch(3XCURSES) | 319 |
| del_curterm(3CURSES) | 320 |
| del_curterm(3XCURSES) | 324 |
| deleteln(3CURSES) | 326 |
| deleteln(3XCURSES) | 327 |
| del_panel(3CURSES) | 328 |
| delscreen(3CURSES) | 329 |
| delscreen(3XCURSES) | 331 |
| delwin(3CURSES) | 332 |
| delwin(3XCURSES) | 335 |
| derwin(3CURSES) | 336 |
| derwin(3XCURSES) | 339 |
| doupdate(3CURSES) | 341 |
| doupdate(3XCURSES) | 343 |
| dup_field(3CURSES) | 344 |
| dupwin(3CURSES) | 345 |
| dupwin(3XCURSES) | 348 |
| dynamic_field_info(3CURSES) | 349 |
| echo(3CURSES) | 350 |
| echo(3XCURSES) | 353 |
| echochar(3CURSES) | 354 |
| echochar(3XCURSES) | 357 |
| echowchar(3CURSES) | 358 |
| echo_wchar(3XCURSES) | 361 |
| endwin(3CURSES) | 362 |
| endwin(3XCURSES) | 364 |
| erase(3CURSES) | 365 |
| erase(3PLOT) | 366 |
| erase(3XCURSES) | 369 |
| erasechar(3CURSES) | 370 |
| erasechar(3XCURSES) | 372 |
| erasewchar(3XCURSES) | 373 |
| field_arg(3CURSES) | 374 |

field_back(3CURSES) 375
field_buffer(3CURSES) 376
field_count(3CURSES) 377
field_fore(3CURSES) 378
field_index(3CURSES) 379
field_info(3CURSES) 381
field_init(3CURSES) 382
field_just(3CURSES) 384
field_opts(3CURSES) 385
field_opts_off(3CURSES) 387
field_opts_on(3CURSES) 389
field_pad(3CURSES) 391
field_status(3CURSES) 392
field_term(3CURSES) 393
field_type(3CURSES) 395
field_userptr(3CURSES) 396
filter(3CURSES) 397
filter(3XCURSES) 399
flash(3CURSES) 400
flash(3XCURSES) 401
flushinp(3CURSES) 402
flushinp(3XCURSES) 404
form_cursor(3CURSES) 405
form_data(3CURSES) 406
form_driver(3CURSES) 407
form_field(3CURSES) 410
form_field_attributes(3CURSES) 411
form_field_buffer(3CURSES) 412
form_field_info(3CURSES) 413
form_field_just(3CURSES) 414
form_field_new(3CURSES) 415
form_field_opts(3CURSES) 416
form_fields(3CURSES) 418
form_fieldtype(3CURSES) 419
form_field_userptr(3CURSES) 421
form_field_validation(3CURSES) 422
form_hook(3CURSES) 423
form_init(3CURSES) 425

| | |
|-------------------------|-----|
| form_new(3CURSES) | 427 |
| form_new_page(3CURSES) | 428 |
| form_opts(3CURSES) | 429 |
| form_opts_off(3CURSES) | 430 |
| form_opts_on(3CURSES) | 431 |
| form_page(3CURSES) | 432 |
| form_post(3CURSES) | 434 |
| forms(3CURSES) | 435 |
| form_sub(3CURSES) | 439 |
| form_term(3CURSES) | 440 |
| form_userptr(3CURSES) | 442 |
| form_win(3CURSES) | 443 |
| free_field(3CURSES) | 444 |
| free_fieldtype(3CURSES) | 445 |
| free_form(3CURSES) | 447 |
| free_item(3CURSES) | 448 |
| free_menu(3CURSES) | 449 |
| getbegyx(3CURSES) | 450 |
| getbegyx(3XCURSES) | 451 |
| getbkgd(3XCURSES) | 452 |
| getbkgrnd(3XCURSES) | 454 |
| getcchar(3XCURSES) | 456 |
| getch(3CURSES) | 457 |
| getch(3XCURSES) | 462 |
| getmaxyx(3CURSES) | 467 |
| getmaxyx(3XCURSES) | 468 |
| getnstr(3XCURSES) | 469 |
| getnwstr(3CURSES) | 471 |
| getn_wstr(3XCURSES) | 472 |
| getparyx(3CURSES) | 474 |
| getparyx(3XCURSES) | 475 |
| getstr(3CURSES) | 476 |
| getstr(3XCURSES) | 477 |
| getsyx(3CURSES) | 479 |
| getwch(3CURSES) | 481 |
| get_wch(3XCURSES) | 486 |
| getwin(3CURSES) | 488 |
| getwin(3XCURSES) | 490 |

getwstr(3CURSES) 491
get_wstr(3XCURSES) 492
getyx(3CURSES) 494
getyx(3XCURSES) 495
halfdelay(3CURSES) 496
halfdelay(3XCURSES) 499
has_colors(3CURSES) 500
has_colors(3XCURSES) 503
has_ic(3CURSES) 506
has_ic(3XCURSES) 508
has_il(3CURSES) 509
has_il(3XCURSES) 511
hide_panel(3CURSES) 512
hline(3XCURSES) 513
hline_set(3XCURSES) 515
idcok(3CURSES) 517
idcok(3XCURSES) 520
idlok(3CURSES) 521
idlok(3XCURSES) 524
immedok(3CURSES) 526
immedok(3XCURSES) 529
inch(3CURSES) 530
inch(3XCURSES) 531
inchnstr(3CURSES) 532
inchnstr(3XCURSES) 533
inchstr(3CURSES) 535
inchstr(3XCURSES) 536
init_color(3CURSES) 538
init_color(3XCURSES) 541
init_pair(3CURSES) 544
init_pair(3XCURSES) 547
initscr(3CURSES) 550
initscr(3XCURSES) 552
innstr(3CURSES) 554
innstr(3XCURSES) 555
innwstr(3CURSES) 557
innwstr(3XCURSES) 558
insch(3CURSES) 560

| | |
|---------------------------|-----|
| insch(3XCURSES) | 561 |
| insdelln(3CURSES) | 562 |
| insdelln(3XCURSES) | 563 |
| insertln(3CURSES) | 564 |
| insertln(3XCURSES) | 565 |
| insnstr(3CURSES) | 566 |
| insnstr(3XCURSES) | 568 |
| insnwstr(3CURSES) | 570 |
| ins_nwstr(3XCURSES) | 572 |
| insstr(3CURSES) | 574 |
| insstr(3XCURSES) | 576 |
| instr(3CURSES) | 578 |
| instr(3XCURSES) | 579 |
| inswch(3CURSES) | 581 |
| ins_wch(3XCURSES) | 582 |
| inswstr(3CURSES) | 584 |
| ins_wstr(3XCURSES) | 586 |
| intrflush(3CURSES) | 588 |
| intrflush(3XCURSES) | 591 |
| inwch(3CURSES) | 592 |
| in_wch(3XCURSES) | 593 |
| inwchnstr(3CURSES) | 594 |
| in_wchnstr(3XCURSES) | 595 |
| inwchstr(3CURSES) | 597 |
| in_wchstr(3XCURSES) | 598 |
| inwstr(3CURSES) | 600 |
| inwstr(3XCURSES) | 601 |
| isendwin(3CURSES) | 603 |
| isendwin(3XCURSES) | 605 |
| is_linetouched(3CURSES) | 606 |
| is_linetouched(3XCURSES) | 608 |
| is_wintouched(3CURSES) | 610 |
| is_wintouched(3XCURSES) | 612 |
| item_count(3CURSES) | 614 |
| item_description(3CURSES) | 615 |
| item_index(3CURSES) | 616 |
| item_init(3CURSES) | 618 |
| item_name(3CURSES) | 620 |

item_opts(3CURSES) 621
item_opts_off(3CURSES) 622
item_opts_on(3CURSES) 623
item_term(3CURSES) 624
item_userptr(3CURSES) 626
item_value(3CURSES) 627
item_visible(3CURSES) 628
keyname(3CURSES) 629
key_name(3XCURSES) 631
keyname(3XCURSES) 633
keypad(3CURSES) 635
keypad(3XCURSES) 638
killchar(3CURSES) 642
killchar(3XCURSES) 644
killwchar(3XCURSES) 645
label(3PLOT) 646
leaveok(3CURSES) 649
leaveok(3XCURSES) 652
libcurses(3XCURSES) 654
line(3PLOT) 662
linemod(3PLOT) 665
LINES(3XCURSES) 668
link_field(3CURSES) 669
link_fieldtype(3CURSES) 670
longname(3CURSES) 672
longname(3XCURSES) 674
menu_attributes(3CURSES) 675
menu_back(3CURSES) 677
menu_cursor(3CURSES) 679
menu_driver(3CURSES) 680
menu_fore(3CURSES) 682
menu_format(3CURSES) 684
menu_grey(3CURSES) 685
menu_hook(3CURSES) 687
menu_init(3CURSES) 689
menu_item_current(3CURSES) 691
menu_item_name(3CURSES) 693
menu_item_new(3CURSES) 694

| | |
|----------------------------|-----|
| menu_item_opts(3CURSES) | 695 |
| menu_items(3CURSES) | 696 |
| menu_item_userptr(3CURSES) | 697 |
| menu_item_value(3CURSES) | 698 |
| menu_item_visible(3CURSES) | 699 |
| menu_mark(3CURSES) | 700 |
| menu_new(3CURSES) | 701 |
| menu_opts(3CURSES) | 702 |
| menu_opts_off(3CURSES) | 703 |
| menu_opts_on(3CURSES) | 704 |
| menu_pad(3CURSES) | 705 |
| menu_pattern(3CURSES) | 707 |
| menu_post(3CURSES) | 708 |
| menus(3CURSES) | 709 |
| menu_sub(3CURSES) | 713 |
| menu_term(3CURSES) | 714 |
| menu_userptr(3CURSES) | 716 |
| menu_win(3CURSES) | 717 |
| meta(3CURSES) | 718 |
| meta(3XCURSES) | 721 |
| move(3CURSES) | 722 |
| move(3PLOT) | 723 |
| move(3XCURSES) | 726 |
| move_field(3CURSES) | 727 |
| movenextch(3CURSES) | 728 |
| move_panel(3CURSES) | 729 |
| moveprevch(3CURSES) | 730 |
| mvaddch(3CURSES) | 731 |
| mvaddch(3XCURSES) | 734 |
| mvaddchnstr(3CURSES) | 736 |
| mvaddchnstr(3XCURSES) | 737 |
| mvaddchstr(3CURSES) | 739 |
| mvaddchstr(3XCURSES) | 740 |
| mvaddnstr(3CURSES) | 742 |
| mvaddnstr(3XCURSES) | 743 |
| mvaddnwstr(3CURSES) | 745 |
| mvaddnwstr(3XCURSES) | 746 |
| mvaddstr(3CURSES) | 748 |

mvaddstr(3XCURSES) 749
mvaddwch(3CURSES) 751
mvadd_wch(3XCURSES) 754
mvaddwchnstr(3CURSES) 756
mvadd_wchnstr(3XCURSES) 758
mvaddwchstr(3CURSES) 760
mvadd_wchstr(3XCURSES) 762
mvaddwstr(3CURSES) 764
mvaddwstr(3XCURSES) 765
mvchgat(3XCURSES) 767
mvcur(3CURSES) 769
mvcur(3XCURSES) 773
mvdelch(3CURSES) 774
mvdelch(3XCURSES) 775
mvderwin(3CURSES) 776
mvderwin(3XCURSES) 779
mvgetch(3CURSES) 780
mvgetch(3XCURSES) 785
mvgetnstr(3XCURSES) 790
mvgetnwstr(3CURSES) 792
mvgetn_wstr(3XCURSES) 793
mvgetstr(3CURSES) 795
mvgetstr(3XCURSES) 796
mvgetwch(3CURSES) 798
mvget_wch(3XCURSES) 803
mvgetwstr(3CURSES) 805
mvget_wstr(3XCURSES) 806
mvhline(3XCURSES) 808
mvhline_set(3XCURSES) 810
mvinch(3CURSES) 812
mvinch(3XCURSES) 813
mvinchnstr(3CURSES) 814
mvinchnstr(3XCURSES) 815
mvinchstr(3CURSES) 817
mvinchstr(3XCURSES) 818
mvinnstr(3CURSES) 820
mvinnstr(3XCURSES) 821
mvinnwstr(3CURSES) 823

| | |
|------------------------|-----|
| mvinnwstr(3XCURSES) | 824 |
| mvinsch(3CURSES) | 826 |
| mvinsch(3XCURSES) | 827 |
| mvinsnstr(3CURSES) | 828 |
| mvinsnstr(3XCURSES) | 830 |
| mvinsnWSTR(3CURSES) | 832 |
| mvins_nwstr(3XCURSES) | 834 |
| mvinsstr(3CURSES) | 836 |
| mvinsstr(3XCURSES) | 838 |
| mvinstr(3CURSES) | 840 |
| mvinstr(3XCURSES) | 841 |
| mvinswch(3CURSES) | 843 |
| mvins_wch(3XCURSES) | 844 |
| mvinswstr(3CURSES) | 846 |
| mvins_wstr(3XCURSES) | 848 |
| mvinwch(3CURSES) | 850 |
| mvin_wch(3XCURSES) | 851 |
| mvinwchnstr(3CURSES) | 852 |
| mvin_wchnstr(3XCURSES) | 853 |
| mvinwchstr(3CURSES) | 855 |
| mvin_wchstr(3XCURSES) | 856 |
| mvinwstr(3CURSES) | 858 |
| mvinwstr(3XCURSES) | 859 |
| mvprintw(3CURSES) | 861 |
| mvprintw(3XCURSES) | 862 |
| mvscanw(3CURSES) | 863 |
| mvscanw(3XCURSES) | 864 |
| mvvline(3XCURSES) | 865 |
| mvvline_set(3XCURSES) | 867 |
| mvwaddch(3CURSES) | 869 |
| mvwaddch(3XCURSES) | 872 |
| mvwaddchnstr(3CURSES) | 874 |
| mvwaddchnstr(3XCURSES) | 875 |
| mvwaddchstr(3CURSES) | 877 |
| mvwaddchstr(3XCURSES) | 878 |
| mvwaddnstr(3CURSES) | 880 |
| mvwaddnstr(3XCURSES) | 881 |
| mvwaddnwstr(3CURSES) | 883 |

mvwaddnwstr(3XCURSES) 884
mvwaddstr(3CURSES) 886
mvwaddstr(3XCURSES) 887
mvwaddwch(3CURSES) 889
mvwadd_wch(3XCURSES) 892
mvwaddwchnstr(3CURSES) 894
mvwadd_wchnstr(3XCURSES) 896
mvwaddwchstr(3CURSES) 898
mvwadd_wchstr(3XCURSES) 900
mvwaddwstr(3CURSES) 902
mvwaddwstr(3XCURSES) 903
mvwchgat(3XCURSES) 905
mvwdelch(3CURSES) 907
mvwdelch(3XCURSES) 908
mvwgetch(3CURSES) 909
mvwgetch(3XCURSES) 914
mvwgetnstr(3XCURSES) 919
mvwgetnwstr(3CURSES) 921
mvwgetn_wstr(3XCURSES) 922
mvwgetstr(3CURSES) 924
mvwgetstr(3XCURSES) 925
mvwgetwch(3CURSES) 927
mvwget_wch(3XCURSES) 932
mvwgetwstr(3CURSES) 934
mvwget_wstr(3XCURSES) 935
mvwhline(3XCURSES) 937
mvwhline_set(3XCURSES) 939
mvwin(3CURSES) 941
mvwin(3XCURSES) 944
mvwinch(3CURSES) 945
mvwinch(3XCURSES) 946
mvwinchnstr(3CURSES) 947
mvwinchnstr(3XCURSES) 948
mvwinchstr(3CURSES) 950
mvwinchstr(3XCURSES) 951
mvwinnstr(3CURSES) 953
mvwinnstr(3XCURSES) 954
mvwinnwstr(3CURSES) 956

| | |
|-------------------------|------|
| mvwinnwstr(3XCURSES) | 957 |
| mvwinsch(3CURSES) | 959 |
| mvwinsch(3XCURSES) | 960 |
| mvwinsnstr(3CURSES) | 961 |
| mvwinsnstr(3XCURSES) | 963 |
| mvwinsnwstr(3CURSES) | 965 |
| mvwins_nwstr(3XCURSES) | 967 |
| mvwinsstr(3CURSES) | 969 |
| mvwinsstr(3XCURSES) | 971 |
| mvwinstr(3CURSES) | 973 |
| mvwinstr(3XCURSES) | 974 |
| mvwinswch(3CURSES) | 976 |
| mvwins_wch(3XCURSES) | 977 |
| mvwinswstr(3CURSES) | 979 |
| mvwins_wstr(3XCURSES) | 981 |
| mvwinwch(3CURSES) | 983 |
| mvwin_wch(3XCURSES) | 984 |
| mvwinwchnstr(3CURSES) | 985 |
| mvwin_wchnstr(3XCURSES) | 986 |
| mvwinwchstr(3CURSES) | 988 |
| mvwin_wchstr(3XCURSES) | 989 |
| mvwinwstr(3CURSES) | 991 |
| mvwinwstr(3XCURSES) | 992 |
| mvwprintw(3CURSES) | 994 |
| mvwprintw(3XCURSES) | 995 |
| mvwscanw(3CURSES) | 996 |
| mvwscanw(3XCURSES) | 997 |
| mvwvline(3XCURSES) | 998 |
| mvwvline_set(3XCURSES) | 1000 |
| napms(3CURSES) | 1002 |
| napms(3XCURSES) | 1004 |
| new_field(3CURSES) | 1005 |
| new_fieldtype(3CURSES) | 1006 |
| new_form(3CURSES) | 1008 |
| new_item(3CURSES) | 1009 |
| new_menu(3CURSES) | 1010 |
| newpad(3CURSES) | 1011 |
| newpad(3XCURSES) | 1013 |

new_page(3CURSES) 1015
new_panel(3CURSES) 1016
newterm(3CURSES) 1017
newterm(3XCURSES) 1019
newwin(3CURSES) 1021
newwin(3XCURSES) 1024
nl(3CURSES) 1026
nl(3XCURSES) 1029
nocbreak(3CURSES) 1030
nocbreak(3XCURSES) 1033
nodelay(3CURSES) 1034
nodelay(3XCURSES) 1037
noecho(3CURSES) 1038
noecho(3XCURSES) 1041
nonl(3CURSES) 1042
nonl(3XCURSES) 1045
noqiflush(3CURSES) 1046
noqiflush(3XCURSES) 1049
noraw(3CURSES) 1050
noraw(3XCURSES) 1053
notimeout(3CURSES) 1054
notimeout(3XCURSES) 1057
openpl(3PLOT) 1059
openvt(3PLOT) 1062
overlay(3CURSES) 1065
overlay(3XCURSES) 1066
overwrite(3CURSES) 1070
overwrite(3XCURSES) 1071
pair_content(3CURSES) 1075
pair_content(3XCURSES) 1078
PAIR_NUMBER(3XCURSES) 1081
panel_above(3CURSES) 1084
panel_below(3CURSES) 1085
panel_hidden(3CURSES) 1086
panel_move(3CURSES) 1087
panel_new(3CURSES) 1088
panels(3CURSES) 1089
panel_show(3CURSES) 1091

| | |
|----------------------------|------|
| panel_top(3CURSES) | 1092 |
| panel_update(3CURSES) | 1093 |
| panel_userptr(3CURSES) | 1094 |
| panel_window(3CURSES) | 1095 |
| pechochar(3CURSES) | 1096 |
| pechochar(3XCURSES) | 1098 |
| pechowchar(3CURSES) | 1099 |
| pecho_wchar(3XCURSES) | 1101 |
| plot(3PLOT) | 1102 |
| pnoutrefresh(3CURSES) | 1105 |
| pnoutrefresh(3XCURSES) | 1107 |
| point(3PLOT) | 1109 |
| pos_form_cursor(3CURSES) | 1112 |
| pos_menu_cursor(3CURSES) | 1113 |
| post_form(3CURSES) | 1114 |
| post_menu(3CURSES) | 1115 |
| prefresh(3CURSES) | 1116 |
| prefresh(3XCURSES) | 1118 |
| printw(3CURSES) | 1120 |
| printw(3XCURSES) | 1121 |
| putp(3CURSES) | 1122 |
| putp(3XCURSES) | 1126 |
| putwin(3CURSES) | 1128 |
| putwin(3XCURSES) | 1130 |
| qiflush(3CURSES) | 1131 |
| qiflush(3XCURSES) | 1134 |
| raw(3CURSES) | 1135 |
| raw(3XCURSES) | 1138 |
| redrawwin(3CURSES) | 1139 |
| redrawwin(3XCURSES) | 1141 |
| refresh(3CURSES) | 1142 |
| refresh(3XCURSES) | 1144 |
| replace_panel(3CURSES) | 1145 |
| reset_prog_mode(3CURSES) | 1146 |
| reset_prog_mode(3XCURSES) | 1148 |
| reset_shell_mode(3CURSES) | 1149 |
| reset_shell_mode(3XCURSES) | 1151 |
| resetty(3CURSES) | 1152 |

resetty(3XCURSES) 1154
restartterm(3CURSES) 1155
restartterm(3XCURSES) 1159
riponline(3CURSES) 1161
riponline(3XCURSES) 1163
savetty(3CURSES) 1164
savetty(3XCURSES) 1166
scale_form(3CURSES) 1167
scale_menu(3CURSES) 1168
scanw(3CURSES) 1169
scanw(3XCURSES) 1170
scr_dump(3CURSES) 1171
scr_dump(3XCURSES) 1173
scr_init(3CURSES) 1174
scr_init(3XCURSES) 1176
sclr(3CURSES) 1177
sclr(3XCURSES) 1178
scroll(3CURSES) 1179
scroll(3XCURSES) 1180
scrollok(3CURSES) 1181
scrollok(3XCURSES) 1184
scr_restore(3CURSES) 1186
scr_restore(3XCURSES) 1188
scr_set(3CURSES) 1189
scr_set(3XCURSES) 1191
setcchar(3XCURSES) 1192
set_current_field(3CURSES) 1193
set_current_item(3CURSES) 1195
set_curterm(3CURSES) 1197
set_curterm(3XCURSES) 1201
set_field_back(3CURSES) 1203
set_field_buffer(3CURSES) 1204
set_field_fore(3CURSES) 1205
set_field_init(3CURSES) 1206
set_field_just(3CURSES) 1208
set_field_opts(3CURSES) 1209
set_field_pad(3CURSES) 1211
set_field_status(3CURSES) 1212

| | |
|-------------------------------|------|
| set_field_term(3CURSES) | 1213 |
| set_field_type(3CURSES) | 1215 |
| set_fieldtype_arg(3CURSES) | 1216 |
| set_fieldtype_choice(3CURSES) | 1218 |
| set_field_userptr(3CURSES) | 1220 |
| set_form_fields(3CURSES) | 1221 |
| set_form_init(3CURSES) | 1222 |
| set_form_opts(3CURSES) | 1224 |
| set_form_page(3CURSES) | 1225 |
| set_form_sub(3CURSES) | 1227 |
| set_form_term(3CURSES) | 1228 |
| set_form_userptr(3CURSES) | 1230 |
| set_form_win(3CURSES) | 1231 |
| set_item_init(3CURSES) | 1232 |
| set_item_opts(3CURSES) | 1234 |
| set_item_term(3CURSES) | 1235 |
| set_item_userptr(3CURSES) | 1237 |
| set_item_value(3CURSES) | 1238 |
| set_max_field(3CURSES) | 1239 |
| set_menu_back(3CURSES) | 1240 |
| set_menu_fore(3CURSES) | 1242 |
| set_menu_format(3CURSES) | 1244 |
| set_menu_grey(3CURSES) | 1245 |
| set_menu_init(3CURSES) | 1247 |
| set_menu_items(3CURSES) | 1249 |
| set_menu_mark(3CURSES) | 1250 |
| set_menu_opts(3CURSES) | 1251 |
| set_menu_pad(3CURSES) | 1252 |
| set_menu_pattern(3CURSES) | 1254 |
| set_menu_sub(3CURSES) | 1255 |
| set_menu_term(3CURSES) | 1256 |
| set_menu_userptr(3CURSES) | 1258 |
| set_menu_win(3CURSES) | 1259 |
| set_new_page(3CURSES) | 1260 |
| set_panel_userptr(3CURSES) | 1261 |
| setscrreg(3CURSES) | 1262 |
| setscrreg(3XCURSES) | 1265 |
| setsyx(3CURSES) | 1267 |

set_term(3CURSES) 1269
setterm(3CURSES) 1271
set_term(3XCURSES) 1275
set_top_row(3CURSES) 1276
setupterm(3CURSES) 1278
setupterm(3XCURSES) 1282
show_panel(3CURSES) 1284
slk_attroff(3CURSES) 1285
slk_attr_off(3XCURSES) 1287
slk_attroff(3XCURSES) 1290
slk_attron(3CURSES) 1293
slk_attr_on(3XCURSES) 1295
slk_attron(3XCURSES) 1298
slk_attrset(3CURSES) 1301
slk_attr_set(3XCURSES) 1303
slk_attrset(3XCURSES) 1306
slk_clear(3CURSES) 1309
slk_clear(3XCURSES) 1311
slk_color(3XCURSES) 1314
slk_init(3CURSES) 1317
slk_init(3XCURSES) 1319
slk_label(3CURSES) 1322
slk_label(3XCURSES) 1324
slk_noutrefresh(3CURSES) 1327
slk_noutrefresh(3XCURSES) 1329
slk_refresh(3CURSES) 1332
slk_refresh(3XCURSES) 1334
slk_restore(3CURSES) 1337
slk_restore(3XCURSES) 1339
slk_set(3CURSES) 1342
slk_set(3XCURSES) 1344
slk_touch(3CURSES) 1347
slk_touch(3XCURSES) 1349
slk_wset(3XCURSES) 1352
space(3PLOT) 1355
standend(3CURSES) 1358
standend(3XCURSES) 1360
standout(3CURSES) 1361

| | |
|-----------------------|------|
| standout(3XCURSES) | 1363 |
| start_color(3CURSES) | 1364 |
| start_color(3XCURSES) | 1367 |
| stdscr(3XCURSES) | 1370 |
| subpad(3CURSES) | 1371 |
| subpad(3XCURSES) | 1373 |
| subwin(3CURSES) | 1375 |
| subwin(3XCURSES) | 1378 |
| syncok(3CURSES) | 1380 |
| syncok(3XCURSES) | 1383 |
| termattrs(3CURSES) | 1384 |
| term_attrs(3XCURSES) | 1386 |
| termattrs(3XCURSES) | 1387 |
| termname(3CURSES) | 1388 |
| termname(3XCURSES) | 1390 |
| tgetent(3CURSES) | 1391 |
| tgetent(3XCURSES) | 1393 |
| tgetflag(3CURSES) | 1395 |
| tgetflag(3XCURSES) | 1397 |
| tgetnum(3CURSES) | 1399 |
| tgetnum(3XCURSES) | 1401 |
| tgetstr(3CURSES) | 1403 |
| tgetstr(3XCURSES) | 1405 |
| tgoto(3CURSES) | 1407 |
| tgoto(3XCURSES) | 1409 |
| tigetflag(3CURSES) | 1411 |
| tigetflag(3XCURSES) | 1415 |
| tigetnum(3CURSES) | 1417 |
| tigetnum(3XCURSES) | 1421 |
| tigetstr(3CURSES) | 1423 |
| tigetstr(3XCURSES) | 1427 |
| timeout(3CURSES) | 1429 |
| timeout(3XCURSES) | 1432 |
| top_panel(3CURSES) | 1434 |
| top_row(3CURSES) | 1435 |
| touchline(3CURSES) | 1437 |
| touchline(3XCURSES) | 1439 |
| touchwin(3CURSES) | 1441 |

touchwin(3XCURSES) 1443
tparm(3CURSES) 1445
tparm(3XCURSES) 1449
tputs(3CURSES) 1451
tputs(3XCURSES) 1455
typeahead(3CURSES) 1457
typeahead(3XCURSES) 1460
unctrl(3CURSES) 1461
unctrl(3XCURSES) 1463
ungetch(3CURSES) 1464
ungetch(3XCURSES) 1469
ungetwch(3CURSES) 1470
unget_wch(3XCURSES) 1475
unpost_form(3CURSES) 1476
unpost_menu(3CURSES) 1477
untouchwin(3CURSES) 1478
untouchwin(3XCURSES) 1480
update_panels(3CURSES) 1482
use_env(3CURSES) 1483
use_env(3XCURSES) 1485
vidattr(3CURSES) 1486
vid_attr(3XCURSES) 1490
vidattr(3XCURSES) 1492
vidputs(3CURSES) 1494
vid_puts(3XCURSES) 1498
vidputs(3XCURSES) 1500
vline(3XCURSES) 1502
vline_set(3XCURSES) 1504
vwprintw(3CURSES) 1506
vw_printw(3XCURSES) 1507
vwprintw(3XCURSES) 1508
vwscanw(3CURSES) 1509
vw_scanw(3XCURSES) 1510
vwscanw(3XCURSES) 1511
waddch(3CURSES) 1512
waddch(3XCURSES) 1515
waddchnstr(3CURSES) 1517
waddchnstr(3XCURSES) 1518

| | |
|------------------------|------|
| waddchstr(3CURSES) | 1520 |
| waddchstr(3XCURSES) | 1521 |
| waddnstr(3CURSES) | 1523 |
| waddnstr(3XCURSES) | 1524 |
| waddnwstr(3CURSES) | 1526 |
| waddnwstr(3XCURSES) | 1527 |
| waddstr(3CURSES) | 1529 |
| waddstr(3XCURSES) | 1530 |
| waddwch(3CURSES) | 1532 |
| wadd_wch(3XCURSES) | 1535 |
| waddwchnstr(3CURSES) | 1537 |
| wadd_wchnstr(3XCURSES) | 1539 |
| waddwchstr(3CURSES) | 1541 |
| wadd_wchstr(3XCURSES) | 1543 |
| waddwstr(3CURSES) | 1545 |
| waddwstr(3XCURSES) | 1546 |
| wadjcurspos(3CURSES) | 1548 |
| wattr_get(3XCURSES) | 1549 |
| wattroff(3CURSES) | 1551 |
| wattr_off(3XCURSES) | 1553 |
| wattroff(3XCURSES) | 1555 |
| wattron(3CURSES) | 1557 |
| wattr_on(3XCURSES) | 1559 |
| wattron(3XCURSES) | 1561 |
| wattrset(3CURSES) | 1563 |
| wattr_set(3XCURSES) | 1565 |
| wattrset(3XCURSES) | 1567 |
| wbkgd(3CURSES) | 1569 |
| wbkgd(3XCURSES) | 1570 |
| wbkgdset(3CURSES) | 1572 |
| wbkgdset(3XCURSES) | 1573 |
| wbkgrnd(3XCURSES) | 1575 |
| wbkgrndset(3XCURSES) | 1577 |
| wborder(3CURSES) | 1579 |
| wborder(3XCURSES) | 1581 |
| wborder_set(3XCURSES) | 1583 |
| wchgat(3XCURSES) | 1586 |
| wclear(3CURSES) | 1588 |

wclear(3XCURSES) 1589
wclrtobot(3CURSES) 1590
wclrtobot(3XCURSES) 1591
wclrtoeol(3CURSES) 1592
wclrtoeol(3XCURSES) 1593
wcolor_set(3XCURSES) 1594
wcursyncup(3CURSES) 1596
wcursyncup(3XCURSES) 1599
wdelch(3CURSES) 1600
wdelch(3XCURSES) 1601
wdeleteln(3CURSES) 1602
wdeleteln(3XCURSES) 1603
wechochar(3CURSES) 1604
wechochar(3XCURSES) 1607
wechowchar(3CURSES) 1608
wecho_wchar(3XCURSES) 1611
werase(3CURSES) 1612
werase(3XCURSES) 1613
wgetbkgrnd(3XCURSES) 1614
wgetch(3CURSES) 1616
wgetch(3XCURSES) 1621
wgetnstr(3CURSES) 1626
wgetnstr(3XCURSES) 1627
wgetnwstr(3CURSES) 1629
wgetn_wstr(3XCURSES) 1630
wgetstr(3CURSES) 1632
wgetstr(3XCURSES) 1633
wgetwch(3CURSES) 1635
wget_wch(3XCURSES) 1640
wgetwstr(3CURSES) 1642
wget_wstr(3XCURSES) 1643
whline(3CURSES) 1645
whline(3XCURSES) 1647
whline_set(3XCURSES) 1649
winch(3CURSES) 1651
winch(3XCURSES) 1652
winchnstr(3CURSES) 1653
winchnstr(3XCURSES) 1654

| | |
|------------------------|------|
| winchstr(3CURSES) | 1656 |
| winchstr(3XCURSES) | 1657 |
| winnstr(3CURSES) | 1659 |
| winnstr(3XCURSES) | 1660 |
| winnwstr(3CURSES) | 1662 |
| winnwstr(3XCURSES) | 1663 |
| winsch(3CURSES) | 1665 |
| winsch(3XCURSES) | 1666 |
| winsdelln(3CURSES) | 1667 |
| winsdelln(3XCURSES) | 1668 |
| winsertln(3CURSES) | 1669 |
| winsertln(3XCURSES) | 1670 |
| winsnstr(3CURSES) | 1671 |
| winsnstr(3XCURSES) | 1673 |
| winsnwstr(3CURSES) | 1675 |
| wins_nwstr(3XCURSES) | 1677 |
| winsstr(3CURSES) | 1679 |
| winsstr(3XCURSES) | 1681 |
| winstr(3CURSES) | 1683 |
| winstr(3XCURSES) | 1684 |
| winswch(3CURSES) | 1686 |
| wins_wch(3XCURSES) | 1687 |
| winswstr(3CURSES) | 1689 |
| wins_wstr(3XCURSES) | 1691 |
| winwch(3CURSES) | 1693 |
| win_wch(3XCURSES) | 1694 |
| winwchnstr(3CURSES) | 1695 |
| win_wchnstr(3XCURSES) | 1696 |
| winwchstr(3CURSES) | 1698 |
| win_wchstr(3XCURSES) | 1699 |
| winwstr(3CURSES) | 1701 |
| winwstr(3XCURSES) | 1702 |
| wmove(3CURSES) | 1704 |
| wmove(3XCURSES) | 1705 |
| wmovenextch(3CURSES) | 1706 |
| wmoveprevch(3CURSES) | 1707 |
| wnoutrefresh(3CURSES) | 1708 |
| wnoutrefresh(3XCURSES) | 1710 |

wprintw(3CURSES) 1711
wprintw(3XCURSES) 1712
wredrawln(3CURSES) 1713
wredrawln(3XCURSES) 1715
wrefresh(3CURSES) 1716
wrefresh(3XCURSES) 1718
wscanw(3CURSES) 1719
wscanw(3XCURSES) 1720
wscl(3CURSES) 1721
wscl(3XCURSES) 1722
wsetscreg(3CURSES) 1723
wsetscreg(3XCURSES) 1726
wstandend(3CURSES) 1728
wstandend(3XCURSES) 1730
wstandout(3CURSES) 1731
wstandout(3XCURSES) 1733
wsyncdown(3CURSES) 1734
wsyncdown(3XCURSES) 1737
wsyncup(3CURSES) 1738
wsyncup(3XCURSES) 1741
wtimeout(3CURSES) 1742
wtimeout(3XCURSES) 1745
wtouchln(3CURSES) 1747
wtouchln(3XCURSES) 1749
wunctrl(3XCURSES) 1751
wvline(3CURSES) 1752
wvline(3XCURSES) 1754
wvline_set(3XCURSES) 1756

Index 1759

Preface

Both novice users and those familiar with the SunOS operating system can use online man pages to obtain information about the system and its features. A man page is intended to answer concisely the question “What does it do?” The man pages in general comprise a reference manual. They are not intended to be a tutorial.

Overview

The following contains a brief description of each man page section and the information it references:

- Section 1 describes, in alphabetical order, commands available with the operating system.
- Section 1M describes, in alphabetical order, commands that are used chiefly for system maintenance and administration purposes.
- Section 2 describes all of the system calls. Most of these calls have one or more error returns. An error condition is indicated by an otherwise impossible returned value.
- Section 3 describes functions found in various libraries, other than those functions that directly invoke UNIX system primitives, which are described in Section 2.
- Section 4 outlines the formats of various files. The C structure declarations for the file formats are given where applicable.
- Section 5 contains miscellaneous documentation such as character-set tables.
- Section 6 contains available games and demos.
- Section 7 describes various special files that refer to specific hardware peripherals and device drivers. STREAMS software drivers, modules and the STREAMS-generic set of system calls are also described.

- Section 9 provides reference information needed to write device drivers in the kernel environment. It describes two device driver interface specifications: the Device Driver Interface (DDI) and the Driver/Kernel Interface (DKI).
- Section 9E describes the DDI/DKI, DDI-only, and DKI-only entry-point routines a developer can include in a device driver.
- Section 9F describes the kernel functions available for use by device drivers.
- Section 9S describes the data structures used by drivers to share information between the driver and the kernel.

Below is a generic format for man pages. The man pages of each manual section generally follow this order, but include only needed headings. For example, if there are no bugs to report, there is no BUGS section. See the `intro` pages for more information and detail about each section, and `man(1)` for more information about man pages in general.

| | |
|----------|--|
| NAME | This section gives the names of the commands or functions documented, followed by a brief description of what they do. |
| SYNOPSIS | This section shows the syntax of commands or functions. When a command or file does not exist in the standard path, its full path name is shown. Options and arguments are alphabetized, with single letter arguments first, and options with arguments next, unless a different argument order is required. |
| | The following special characters are used in this section: |
| [] | Brackets. The option or argument enclosed in these brackets is optional. If the brackets are omitted, the argument must be specified. |
| . . . | Ellipses. Several values can be provided for the previous argument, or the previous argument can be specified multiple times, for example, "filename . . .". |
| | Separator. Only one of the arguments separated by this character can be specified at a time. |
| { } | Braces. The options and/or arguments enclosed within braces are interdependent, such that everything enclosed must be treated as a unit. |

| | |
|---------------|--|
| PROTOCOL | This section occurs only in subsection 3R to indicate the protocol description file. |
| DESCRIPTION | This section defines the functionality and behavior of the service. Thus it describes concisely what the command does. It does not discuss OPTIONS or cite EXAMPLES. Interactive commands, subcommands, requests, macros, and functions are described under USAGE. |
| IOCTL | This section appears on pages in Section 7 only. Only the device class that supplies appropriate parameters to the <code>ioctl(2)</code> system call is called <code>ioctl</code> and generates its own heading. <code>ioctl</code> calls for a specific device are listed alphabetically (on the man page for that specific device). <code>ioctl</code> calls are used for a particular class of devices all of which have an <code>io</code> ending, such as <code>mtio(7I)</code> . |
| OPTIONS | This section lists the command options with a concise summary of what each option does. The options are listed literally and in the order they appear in the SYNOPSIS section. Possible arguments to options are discussed under the option, and where appropriate, default values are supplied. |
| OPERANDS | This section lists the command operands and describes how they affect the actions of the command. |
| OUTPUT | This section describes the output – standard output, standard error, or output files – generated by the command. |
| RETURN VALUES | If the man page documents functions that return values, this section lists these values and describes the conditions under which they are returned. If a function can return only constant values, such as 0 or -1, these values are listed in tagged paragraphs. Otherwise, a single paragraph describes the return values of each function. Functions declared void do not return values, so they are not discussed in RETURN VALUES. |
| ERRORS | On failure, most functions place an error code in the global variable <code>errno</code> indicating why they failed. This section lists alphabetically all error codes a function can generate and describes the conditions that cause each error. When more than |

| | |
|-----------------------|--|
| | one condition can cause the same error, each condition is described in a separate paragraph under the error code. |
| USAGE | This section lists special rules, features, and commands that require in-depth explanations. The subsections listed here are used to explain built-in functionality: Commands Modifiers Variables Expressions Input Grammar |
| EXAMPLES | This section provides examples of usage or of how to use a command or function. Wherever possible a complete example including command-line entry and machine response is shown. Whenever an example is given, the prompt is shown as <code>example%</code> , or if the user must be superuser, <code>example#</code> . Examples are followed by explanations, variable substitution rules, or returned values. Most examples illustrate concepts from the SYNOPSIS, DESCRIPTION, OPTIONS, and USAGE sections. |
| ENVIRONMENT VARIABLES | This section lists any environment variables that the command or function affects, followed by a brief description of the effect. |
| EXIT STATUS | This section lists the values the command returns to the calling program or shell and the conditions that cause these values to be returned. Usually, zero is returned for successful completion, and values other than zero for various error conditions. |
| FILES | This section lists all file names referred to by the man page, files of interest, and files created or required by commands. Each is followed by a descriptive summary or explanation. |
| ATTRIBUTES | This section lists characteristics of commands, utilities, and device drivers by defining the attribute type and its corresponding value. See <code>attributes(5)</code> for more information. |
| SEE ALSO | This section lists references to other man pages, in-house documentation, and outside publications. |

| | |
|-------------|---|
| DIAGNOSTICS | This section lists diagnostic messages with a brief explanation of the condition causing the error. |
| WARNINGS | This section lists warnings about special conditions which could seriously affect your working conditions. This is not a list of diagnostics. |
| NOTES | This section lists additional information that does not belong anywhere else on the page. It takes the form of an aside to the user, covering points of special interest. Critical information is never covered here. |
| BUGS | This section describes known bugs and, wherever possible, suggests workarounds. |

Curses Library Functions

addch(3CURSES)

| | |
|----------------------|--|
| NAME | <code> curs_addch, addch, waddch, mvaddch, mvwaddch, echochar, wechochar </code> – add a character (with attributes) to a curses window and advance cursor |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int addch(chtype <i>ch</i>); int waddch(WINDOW *<i>win</i>, chtype <i>ch</i>); int mvaddch(int <i>y</i>, int <i>x</i>, chtype <i>ch</i>); int mvwaddch(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>, chtype <i>ch</i>); int echochar(chtype <i>ch</i>); int wechochar(WINDOW *<i>win</i>, chtype <i>ch</i>);</pre> |
| DESCRIPTION | <p>With the <code>addch()</code>, <code>waddch()</code>, <code>mvaddch()</code>, and <code>mvwaddch()</code> routines, the character <i>ch</i> is put into the window at the current cursor position of the window and the position of the window cursor is advanced. Its function is similar to that of <code>putchar()</code>. At the right margin, an automatic newline is performed. At the bottom of the scrolling region, if <code>scrollok()</code> is enabled, the scrolling region is scrolled up one line.</p> <p>If <i>ch</i> is a tab, newline, or backspace, the cursor is moved appropriately within the window. A newline also does a <code>clrtoeol()</code> before moving. Tabs are considered to be at every eighth column. If <i>ch</i> is another control character, it is drawn in the <code>^X</code> notation. Calling <code>winch()</code> after adding a control character does not return the control character, but instead returns the representation of the control character. See <code>curs_inch(3CURSES)</code>.</p> <p>Video attributes can be combined with a character by OR-ing them into the parameter. This results in these attributes also being set. (The intent here is that text, including attributes, can be copied from one place to another using <code>inch()</code> and <code>addch()</code>.) (see <code>standout()</code>, predefined video attribute constants, on the <code>curs_attr(3CURSES)</code> page).</p> <p>The <code>echochar()</code> and <code>wechochar()</code> routines are functionally equivalent to a call to <code>addch()</code> followed by a call to <code>refresh()</code>, or a call to <code>waddch</code> followed by a call to <code>wrefresh()</code>. The knowledge that only a single character is being output is taken into consideration and, for non-control characters, a considerable performance gain might be seen by using these routines instead of their equivalents.</p> |
| Line Graphics | The following variables may be used to add line drawing characters to the screen with routines of the <code>addch()</code> family. When variables are defined for the terminal, the <code>A_ALTCHARSET</code> bit is turned on (see <code>curs_attr(3CURSES)</code>). Otherwise, the default character listed below is stored in the variable. The names chosen are consistent with the VT100 nomenclature. |

addch(3CURSES)

| Name | Default | Glyph Description |
|--------------|---------|-------------------------|
| ACS_ULCORNER | + | upper left-hand corner |
| ACS_LLCORNER | + | lower left-hand corner |
| ACS_URCORNER | + | upper right-hand corner |
| ACS_LRCORNER | + | lower right-hand corner |
| ACS_RTEE | + | right tee |
| ACS_LTEE | + | left tee |
| ACS_BTEE | + | bottom tee |
| ACS_TTEE | + | top tee |
| ACS_HLINE | - | horizontal line |
| ACS_VLINE | | vertical line |
| ACS_PLUS | + | plus |
| ACS_S1 | - | scan line 1 |
| ACS_S9 | - | scan line 9 |
| ACS_DIAMOND | + | diamond |
| ACS_CKBOARD | : | checker board (stipple) |
| ACS_DEGREE | ' | degree symbol |
| ACS_PLMINUS | # | plus/minus |
| ACS_BULLET | o | bullet |
| ACS_LARROW | < | arrow pointing left |
| ACS_RARROW | > | arrow pointing right |
| ACS_DARROW | v | arrow pointing down |
| ACS_UARROW | ^ | arrow pointing up |
| ACS_BOARD | # | board of squares |
| ACS_LANTERN | # | lantern symbol |
| ACS_BLOCK | # | solid square block |

RETURN VALUES

All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

addch(3CURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_attr(3CURSES)`, `curl_clear(3CURSES)`, `curl_inch(3CURSES)`, `curl_outopts(3CURSES)`, `curl_refresh(3CURSES)`, `curses(3CURSES)`, `putc(3C)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `addch()`, `mvaddch()`, `mvwaddch()`, and `echochar()` may be macros.

| | | | | | | | | | | | |
|----------------------|---|---------------|---|----------|--|----------|---|----------|--|------------|---|
| NAME | addch, mvaddch, mvwaddch, waddch – add a character (with rendition) to a window | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addch(const chtype ch); int mvaddch(int y, int x, const chtype ch); int mvwaddch(WINDOW *win, int y, int x, const chtype ch); int waddch(WINDOW *win, const chtype ch);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The <code>addch()</code> function writes a character to the <code>stdscr</code> window at the current cursor position. The <code>mvaddch()</code> and <code>mvwaddch()</code> functions write the character to the position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters. The <code>mvaddch()</code> function writes the character to the <code>stdscr</code> window, while <code>mvwaddch()</code> writes the character to the window specified by <code>win</code>. The <code>waddch()</code> function is identical to <code>addch()</code>, but writes the character to the window specified by <code>win</code>.</p> <p>These functions advance the cursor after writing the character. Characters that do not fit on the end of the current line are wrapped to the beginning of the next line unless the current line is the last line of the window and scrolling is disabled. In that situation, characters which extend beyond the end of the line are discarded.</p> <p>When <code>ch</code> is a backspace, carriage return, newline, or tab, X/Open Curses moves the cursor appropriately. Each tab character moves the cursor to the next tab stop. By default, tab stops occur every eight columns. When <code>ch</code> is a control character other than backspace, carriage return, newline, or tab, it is written using <code>^x</code> notation, where <code>x</code> is a printable character. When X/Open Curses writes <code>ch</code> to the last character position on a line, it automatically generates a newline. When <code>ch</code> is written to the last character position of a scrolling region and <code>scrollok()</code> is enabled, X/Open Curses scrolls the scrolling region up one line (see <code>clearok(3XCURSES)</code>).</p> | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 10px;"><i>wchstr</i></td> <td>Is a pointer to the <code>cchar_t</code> string to be copied to the window.</td> </tr> <tr> <td style="padding-right: 10px;"><i>n</i></td> <td>Is the maximum number of characters to be copied from <i>wchstr</i>. If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line.</td> </tr> <tr> <td style="padding-right: 10px;"><i>y</i></td> <td>Is the <code>y</code> (row) coordinate of the starting position of <i>wchstr</i> in the window.</td> </tr> <tr> <td style="padding-right: 10px;"><i>x</i></td> <td>Is the <code>x</code> (column) coordinate of the starting position of <i>wchstr</i> in the window.</td> </tr> <tr> <td style="padding-right: 10px;"><i>win</i></td> <td>Is a pointer to the window to which the string is to be copied.</td> </tr> </table> | <i>wchstr</i> | Is a pointer to the <code>cchar_t</code> string to be copied to the window. | <i>n</i> | Is the maximum number of characters to be copied from <i>wchstr</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | <i>y</i> | Is the <code>y</code> (row) coordinate of the starting position of <i>wchstr</i> in the window. | <i>x</i> | Is the <code>x</code> (column) coordinate of the starting position of <i>wchstr</i> in the window. | <i>win</i> | Is a pointer to the window to which the string is to be copied. |
| <i>wchstr</i> | Is a pointer to the <code>cchar_t</code> string to be copied to the window. | | | | | | | | | | |
| <i>n</i> | Is the maximum number of characters to be copied from <i>wchstr</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | | | | | | | | | | |
| <i>y</i> | Is the <code>y</code> (row) coordinate of the starting position of <i>wchstr</i> in the window. | | | | | | | | | | |
| <i>x</i> | Is the <code>x</code> (column) coordinate of the starting position of <i>wchstr</i> in the window. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window to which the string is to be copied. | | | | | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | | | | | |

addch(3XCURSES)

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attroff(3XCURSES)`, `bkgdset(3XCURSES)`, `doupage(3XCURSES)`, `inch(3XCURSES)`, `insch(3XCURSES)`, `libcurses(3XCURSES)`, `nl(3XCURSES)`, `printw(3XCURSES)`, `scrollok(3XCURSES)`, `scl(3XCURSES)`, `terminfo(4)`, `attributes(5)`, `standards(5)`

| NAME | curs_addchstr, addchstr, addchnstr, waddchstr, waddchnstr, mvaddchstr, mvaddchnstr, mvwaddchstr, mvwaddchnstr – add string of characters and attributes to a curses window | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int addchstr(chtype *chstr); int addchnstr(chtype *chstr, int n); int waddchstr(WINDOW *win, chtype *chstr); int waddchnstr(WINDOW *win, chtype *chstr, int n); int mvaddchstr(int y, int x, chtype *chstr); int mvaddchnstr(int y, int x, chtype *chstr, int n); int mvwaddchstr(WINDOW *win, int y, int x, chtype *chstr); int mvwaddchnstr(WINDOW *win, int y, int x, chtype *chstr, int n);</pre> | | | | |
| DESCRIPTION | <p>All of these routines copy <i>chstr</i> directly into the window image structure starting at the current cursor position. The four routines with <i>n</i> as the last argument copy at most <i>n</i> elements, but no more than will fit on the line. If <i>n</i> = -1 then the whole string is copied, to the maximum number that fit on the line.</p> <p>The position of the window cursor is not advanced. These routines works faster than <code>waddnstr()</code> (see <code>curs_addstr(3CURSES)</code>) because they merely copy <i>chstr</i> into the window image structure. On the other hand, care must be taken when using these functions because they do not perform any kind of checking (such as for the newline character), they do not advance the current cursor position, and they truncate the string, rather than wrapping it around to the next line.</p> | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_addstr(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that all routines except <code>waddchnstr()</code> and <code>waddchstr()</code> may be macros.</p> | | | | |

addchnstr(3XCURSES)

| | | | | | | | | | | | |
|----------------------|---|--------------------|--|----------------|---|----------------|--|----------------|---|------------------|---|
| NAME | addchstr, addchnstr, mvaddchstr, mvaddchnstr, mvwaddchstr, mvwaddchnstr, waddchstr, waddchnstr – copy a character string (with renditions) to a window | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addchstr(const chtype *chstr); int addchnstr(const chtype *chstr, int n); int mvaddchnstr(int y, int x, const chtype *chstr, int n); int mvaddchstr(int y, int x, const chtype *chstr); int mvwaddchnstr(WINDOW *win, int y, int x, const chtype *chstr, int n); int mvwaddchstr(WINDOW *win, int y, int x, const chtype *chstr); int waddchstr(WINDOW *win, const chtype *chstr); int waddchnstr(WINDOW *win, const chtype *chstr, int n);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The <code>addchstr()</code> function copies the <code>chtype</code> character string to the <code>stdscr</code> window at the current cursor position. The <code>mvaddchstr()</code> and <code>mvwaddchstr()</code> functions copy the character string to the starting position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <code>win</code>). The <code>waddchstr()</code> is identical to <code>addchstr()</code>, but writes to the window specified by <code>win</code>.</p> <p>The <code>addchnstr()</code>, <code>waddchnstr()</code>, <code>mvaddchnstr()</code>, and <code>mvwaddchnstr()</code> functions write <code>n</code> characters to the window, or as many as will fit on the line. If <code>n</code> is less than 0, the entire string is written, or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <code>x</code> and <code>y</code> parameters.</p> <p>These functions differ from the <code>addstr(3XCURSES)</code> set of functions in two important respects. First, these functions do <i>not</i> advance the cursor after writing the string to the window. Second, the current window rendition is not combined with the character; only the attributes that are already part of the <code>chtype</code> character are used.</p> | | | | | | | | | | |
| PARAMETERS | <table><tr><td><code>chstr</code></td><td>Is a pointer to the <code>chtype</code> string to be copied to the window.</td></tr><tr><td><code>n</code></td><td>Is the maximum number of characters to be copied from <code>chstr</code>. If <code>n</code> is less than 0, the entire string is written or as much of it as fits on the line.</td></tr><tr><td><code>y</code></td><td>Is the <code>y</code> (row) coordinate of the starting position of <code>chstr</code> in the window.</td></tr><tr><td><code>x</code></td><td>Is the <code>x</code> (column) coordinate of the starting position of <code>chstr</code> in the window.</td></tr><tr><td><code>win</code></td><td>Is a pointer to the window to which the string is to be copied.</td></tr></table> | <code>chstr</code> | Is a pointer to the <code>chtype</code> string to be copied to the window. | <code>n</code> | Is the maximum number of characters to be copied from <code>chstr</code> . If <code>n</code> is less than 0, the entire string is written or as much of it as fits on the line. | <code>y</code> | Is the <code>y</code> (row) coordinate of the starting position of <code>chstr</code> in the window. | <code>x</code> | Is the <code>x</code> (column) coordinate of the starting position of <code>chstr</code> in the window. | <code>win</code> | Is a pointer to the window to which the string is to be copied. |
| <code>chstr</code> | Is a pointer to the <code>chtype</code> string to be copied to the window. | | | | | | | | | | |
| <code>n</code> | Is the maximum number of characters to be copied from <code>chstr</code> . If <code>n</code> is less than 0, the entire string is written or as much of it as fits on the line. | | | | | | | | | | |
| <code>y</code> | Is the <code>y</code> (row) coordinate of the starting position of <code>chstr</code> in the window. | | | | | | | | | | |
| <code>x</code> | Is the <code>x</code> (column) coordinate of the starting position of <code>chstr</code> in the window. | | | | | | | | | | |
| <code>win</code> | Is a pointer to the window to which the string is to be copied. | | | | | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | | | | | |

addchnstr(3XCURSES)

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addch(3XCURSES)`, `addnstr(3XCURSES)`, `attroff(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

addchstr(3CURSES)

| NAME | <code> curs_addchstr, addchstr, addchnstr, waddchstr, waddchnstr, mvaddchstr, mvaddchnstr, mvwaddchstr, mvwaddchnstr</code> – add string of characters and attributes to a curses window | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int addchstr(chtype *chstr); int addchnstr(chtype *chstr, int n); int waddchstr(WINDOW *win, chtype *chstr); int waddchnstr(WINDOW *win, chtype *chstr, int n); int mvaddchstr(int y, int x, chtype *chstr); int mvaddchnstr(int y, int x, chtype *chstr, int n); int mvwaddchstr(WINDOW *win, int y, int x, chtype *chstr); int mvwaddchnstr(WINDOW *win, int y, int x, chtype *chstr, int n);</pre> | | | | |
| DESCRIPTION | <p>All of these routines copy <i>chstr</i> directly into the window image structure starting at the current cursor position. The four routines with <i>n</i> as the last argument copy at most <i>n</i> elements, but no more than will fit on the line. If <i>n</i>=-1 then the whole string is copied, to the maximum number that fit on the line.</p> <p>The position of the window cursor is not advanced. These routines works faster than <code>waddnstr()</code> (see <code>curs_addstr(3CURSES)</code>) because they merely copy <i>chstr</i> into the window image structure. On the other hand, care must be taken when using these functions because they do not perform any kind of checking (such as for the newline character), they do not advance the current cursor position, and they truncate the string, rather than wrapping it around to the next line.</p> | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_addstr(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that all routines except <code>waddchnstr()</code> and <code>waddchstr()</code> may be macros.</p> | | | | |

| | |
|----------------------|---|
| NAME | addchstr, addchnstr, mvaddchstr, mvaddchnstr, mvwaddchnstr, mvwaddchstr, waddchstr, waddchnstr – copy a character string (with renditions) to a window |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addchstr(const chtype *chstr); int addchnstr(const chtype *chstr, int n); int mvaddchnstr(int y, int x, const chtype *chstr, int n); int mvaddchstr(int y, int x, const chtype *chstr); int mvwaddchnstr(WINDOW *win, int y, int x, const chtype *chstr, int n); int mvwaddchstr(WINDOW *win, int y, int x, const chtype *chstr); int waddchstr(WINDOW *win, const chtype *chstr); int waddchnstr(WINDOW *win, const chtype *chstr, int n);</pre> |
| DESCRIPTION | <p>The <code>addchstr()</code> function copies the <code>chtype</code> character string to the <code>stdscr</code> window at the current cursor position. The <code>mvaddchstr()</code> and <code>mvwaddchstr()</code> functions copy the character string to the starting position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <code>win</code>). The <code>waddchstr()</code> is identical to <code>addchstr()</code>, but writes to the window specified by <code>win</code>.</p> <p>The <code>addchnstr()</code>, <code>waddchnstr()</code>, <code>mvaddchnstr()</code>, and <code>mvwaddchnstr()</code> functions write <code>n</code> characters to the window, or as many as will fit on the line. If <code>n</code> is less than 0, the entire string is written, or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <code>x</code> and <code>y</code> parameters.</p> <p>These functions differ from the <code>addstr(3XCURSES)</code> set of functions in two important respects. First, these functions do <i>not</i> advance the cursor after writing the string to the window. Second, the current window rendition is not combined with the character; only the attributes that are already part of the <code>chtype</code> character are used.</p> |
| PARAMETERS | <p><code>chstr</code> Is a pointer to the <code>chtype</code> string to be copied to the window.</p> <p><code>n</code> Is the maximum number of characters to be copied from <code>chstr</code>. If <code>n</code> is less than 0, the entire string is written or as much of it as fits on the line.</p> <p><code>y</code> Is the <code>y</code> (row) coordinate of the starting position of <code>chstr</code> in the window.</p> <p><code>x</code> Is the <code>x</code> (column) coordinate of the starting position of <code>chstr</code> in the window.</p> <p><code>win</code> Is a pointer to the window to which the string is to be copied.</p> |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. |

addchstr(3XCURSES)

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addch(3XCURSES)`, `addnstr(3XCURSES)`, `attroff(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | <p><code> curs_addstr, addstr, addnstr, waddstr, waddnstr, mvaddstr, mvaddnstr, mvwaddstr, mvwaddnstr</code> – add a string of characters to a curses window and advance cursor</p> | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int addstr(char *str); int addnstr(char *str, int n); int waddstr(WINDOW *win, char *str); int waddnstr(WINDOW *win, char *str, int n); int mvaddstr(int y, int x, char *str); int mvaddnstr(int y, int x, char *str, int n); int mvwaddstr(WINDOW *win, int y, int x, char *str); int mvwaddnstr(WINDOW *win, int y, int x, char *str, int n);</pre> | | | | |
| DESCRIPTION | <p>All of these routines write all the characters of the null terminated character string <i>str</i> on the given window. It is similar to calling <code>waddch()</code> once for each character in the string. The four routines with <i>n</i> as the last argument write at most <i>n</i> characters. If <i>n</i> is negative, then the entire string will be added.</p> | | | | |
| RETURN VALUES | <p>All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion.</p> | | | | |
| ATTRIBUTES | <p>See <code>attributes(5)</code> for descriptions of the following attributes:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <p><code>curs_addch(3CURSES)</code>, <code>curses(3CURSES)</code>, <code>attributes(5)</code></p> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that all routines except <code>waddstr()</code> and <code>waddnstr()</code> may not be macros.</p> | | | | |

addnstr(3XCURSES)

| | | | | | | | |
|--------------------|---|------------|---|----------|---|----------|--|
| NAME | addnstr, addstr, mvaddnstr, mvaddstr, mvwaddnstr, mvwaddstr, waddnstr, waddstr – add a multi-byte character string (without rendition) to a window | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addnstr(const char *str, int n); int addstr(const char *str); int mvaddnstr(int y, int x, const char *str, int n); int mvaddstr(int y, int x, const char *str); int mvwaddnstr(WINDOW *win, int y, int x, const char *str, int n); int mvwaddstr(WINDOW *win, int y, int x, const char *str); int waddstr(WINDOW *win, const char *str); int waddnstr(WINDOW *win, const char *str, int n);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>addstr()</code> function writes a null-terminated string of multi-byte characters to the <code>stdscr</code> window at the current cursor position. The <code>waddstr()</code> function performs an identical action, but writes the character to the window specified by <code>win</code>. The <code>mvaddstr()</code> and <code>mvwaddstr()</code> functions write the string to the position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <code>win</code>).</p> <p>The <code>addnstr()</code>, <code>waddnstr()</code>, <code>mvaddnstr()</code>, and <code>mvwaddnstr()</code> functions are similar but write at most <code>n</code> characters to the window. If <code>n</code> is less than 0, the entire string is written.</p> <p>All of these functions advance the cursor after writing the string.</p> <p>These functions are functionally equivalent to calling the corresponding function from the <code>addch(3XCURSES)</code> set of functions once for each character in the string. Refer to the <code>curses(3XCURSES)</code> man page for a complete description of special character handling and of the interaction between the window rendition (or background character and rendition) and the character written.</p> <p>Note that these functions differ from the <code>addchstr()</code> set of functions in that the <code>addchstr(3XCURSES)</code> functions copy the string as is (without combining each character with the window rendition or the background character and rendition).</p> | | | | | | |
| PARAMETERS | <table><tr><td><i>str</i></td><td>Is a pointer to the character string that is to be written to the window.</td></tr><tr><td><i>n</i></td><td>Is the maximum number of characters to be copied from <i>str</i>. If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line.</td></tr><tr><td><i>y</i></td><td>Is the <i>y</i> (row) coordinate of the starting position of <i>str</i> in the window.</td></tr></table> | <i>str</i> | Is a pointer to the character string that is to be written to the window. | <i>n</i> | Is the maximum number of characters to be copied from <i>str</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of <i>str</i> in the window. |
| <i>str</i> | Is a pointer to the character string that is to be written to the window. | | | | | | |
| <i>n</i> | Is the maximum number of characters to be copied from <i>str</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | | | | | | |
| <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of <i>str</i> in the window. | | | | | | |

addnstr(3XCURSES)

x Is the *x* (column) coordinate of the starting position of *str* in the window.

win Is a pointer to the window in which the string is to be written.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addch(3XCURSES)`, `addchstr(3XCURSES)`, `curses(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

addnwstr(3CURSES)

NAME | curs_addwstr, addwstr, addnwstr, waddwstr, waddnwstr, mvaddwstr, mvaddnwstr, mvwaddwstr, mvwaddnwstr – add a string of wchar_t characters to a curses window and advance cursor

SYNOPSIS | **cc** [flag...] file... -lcurses [library...]

```
#include<curses.h>
int addwstr(wchar_t *wstr);
int addnwstr(wchar_t *wstr, int n);
int waddwstr(WINDOW *win, wchar_t *wstr);
int waddnwstr(WINDOW *win, wchar_t *wstr, int n);
int mvaddwstr(int y, int x, wchar_t *wstr);
int mvaddnwstr(int y, int x, wchar_t *wstr, int n);
int mvwaddwstr(WINDOW *win, int y, int x, wchar_t *wstr);
int mvwaddnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);
```

DESCRIPTION | All of these routines write all the characters of the null-terminated wchar_t character string wstr on the given window. The effect is similar to calling waddwch(3CURSES) once for each wchar_t character in the string. The four routines with n as the last argument write at most n wchar_t characters. If n is negative, then the entire string will be added.

RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | curses(3CURSES), waddwch(3CURSES), attributes(5)

NOTES | The header file <curses.h> automatically includes the header files <stdio.h>, <nctrl.h> and <widec.h>.

Note that all of these routines except waddwstr() and waddnwstr() may be macros.

| | |
|--------------------|---|
| NAME | addnwstr, addwstr, mvaddnwstr, mvaddwstr, mvwaddnwstr, mvwaddwstr, waddnwstr, waddwstr – add a wide-character string to a window |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addnwstr(const wchar_t *wstr, int n); int addwstr(const wchar_t *wstr); int mvaddnwstr(int y, int x, const wchar_t *wstr, int n); int mvaddwstr(int y, int x, const wchar_t *wstr); int mvwaddnwstr(WINDOW*win, int y, int x, const wchar_t *wstr, int n); int mvwaddwstr(WINDOW*win, int y, int x, const wchar_t *wstr); int waddnwstr(WINDOW*win, const wchar_t *wstr, int n); int waddwstr(WINDOW*win, const wchar_t *wstr);</pre> |
| DESCRIPTION | <p>The <code>addwstr()</code> function writes a null-terminated wide-character string to the <code>stdscr</code> window at the current cursor position. The <code>waddwstr()</code> function performs an identical action, but writes the string to the window specified by <code>win</code>. The <code>mvaddwstr()</code> and <code>mvwaddwstr()</code> functions write the string to the position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <code>win</code>).</p> <p>The <code>addnwstr()</code>, <code>waddnwstr()</code>, <code>mvaddnwstr()</code>, and <code>mvwaddnwstr()</code> functions write at most <code>n</code> characters to the window. If <code>n</code> is less than 0, the entire string is written. The former two functions place the characters at the current cursor position; the latter two commands use the position specified by the <code>x</code> and <code>y</code> parameters.</p> <p>All of these functions advance the cursor after writing the string.</p> <p>These functions are functionally equivalent to building a <code>cchar_t</code> from the <code>wchar_t</code> and the window rendition (or background character and rendition) and calling the <code>wadd_wch(3XCURSES)</code> function once for each <code>wchar_t</code> in the string. Refer to the <code>curses(3XCURSES)</code> man page for a complete description of special character handling and of the interaction between the window rendition (or background character and rendition) and the character written.</p> <p>Note that these functions differ from the <code>add_wchnstr(3XCURSES)</code> set of functions in that the latter copy the string as is (without combining each character with the foreground and background attributes of the window).</p> |
| PARAMETERS | <p><code>wstr</code> Is a pointer to the wide-character string that is to be written to the window.</p> |

addnwstr(3XCURSES)

- n* Is the maximum number of characters to be copied from *wstr*. If *n* is less than 0, the entire string is written or as much of it as fits on the line.
- y* Is the y (row) coordinate of the starting position of *wstr* in the window.
- x* Is the x (column) coordinate of the starting position of *wstr* in the window.
- win* Is a pointer to the window in which the string is to be written.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `add_wch(3XCURSES)`, `add_wchnstr(3XCURSES)`, `curses(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | <p><code> curs_addstr, addstr, addnstr, waddstr, waddnstr, mvaddstr, mvaddnstr, mvwaddstr, mvwaddnstr</code> – add a string of characters to a curses window and advance cursor</p> | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int addstr(char *str); int addnstr(char *str, int n); int waddstr(WINDOW *win, char *str); int waddnstr(WINDOW *win, char *str, int n); int mvaddstr(int y, int x, char *str); int mvaddnstr(int y, int x, char *str, int n); int mvwaddstr(WINDOW *win, int y, int x, char *str); int mvwaddnstr(WINDOW *win, int y, int x, char *str, int n);</pre> | | | | |
| DESCRIPTION | <p>All of these routines write all the characters of the null terminated character string <i>str</i> on the given window. It is similar to calling <code>waddch()</code> once for each character in the string. The four routines with <i>n</i> as the last argument write at most <i>n</i> characters. If <i>n</i> is negative, then the entire string will be added.</p> | | | | |
| RETURN VALUES | <p>All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion.</p> | | | | |
| ATTRIBUTES | <p>See <code>attributes(5)</code> for descriptions of the following attributes:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <p><code>curs_addch(3CURSES)</code>, <code>curses(3CURSES)</code>, <code>attributes(5)</code></p> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that all routines except <code>waddstr()</code> and <code>waddnstr()</code> may not be macros.</p> | | | | |

addstr(3XCURSES)

| | | | | | | | |
|--------------------|---|------------|---|----------|---|----------|--|
| NAME | <code>addnstr</code> , <code>addstr</code> , <code>mvaddnstr</code> , <code>mvaddstr</code> , <code>mvwaddnstr</code> , <code>mvwaddstr</code> , <code>waddnstr</code> , <code>waddstr</code> – add a multi-byte character string (without rendition) to a window | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addnstr(const char *str, int n); int addstr(const char *str); int mvaddnstr(int y, int x, const char *str, int n); int mvaddstr(int y, int x, const char *str); int mvwaddnstr(WINDOW *win, int y, int x, const char *str, int n); int mvwaddstr(WINDOW *win, int y, int x, const char *str); int waddstr(WINDOW *win, const char *str); int waddnstr(WINDOW *win, const char *str, int n);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>addstr()</code> function writes a null-terminated string of multi-byte characters to the <code>stdscr</code> window at the current cursor position. The <code>waddstr()</code> function performs an identical action, but writes the character to the window specified by <i>win</i>. The <code>mvaddstr()</code> and <code>mvwaddstr()</code> functions write the string to the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <i>win</i>).</p> <p>The <code>addnstr()</code>, <code>waddnstr()</code>, <code>mvaddnstr()</code>, and <code>mvwaddnstr()</code> functions are similar but write at most <i>n</i> characters to the window. If <i>n</i> is less than 0, the entire string is written.</p> <p>All of these functions advance the cursor after writing the string.</p> <p>These functions are functionally equivalent to calling the corresponding function from the <code>addch(3XCURSES)</code> set of functions once for each character in the string. Refer to the <code>curses(3XCURSES)</code> man page for a complete description of special character handling and of the interaction between the window rendition (or background character and rendition) and the character written.</p> <p>Note that these functions differ from the <code>addchstr()</code> set of functions in that the <code>addchstr(3XCURSES)</code> functions copy the string as is (without combining each character with the window rendition or the background character and rendition).</p> | | | | | | |
| PARAMETERS | <table><tr><td><i>str</i></td><td>Is a pointer to the character string that is to be written to the window.</td></tr><tr><td><i>n</i></td><td>Is the maximum number of characters to be copied from <i>str</i>. If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line.</td></tr><tr><td><i>y</i></td><td>Is the <i>y</i> (row) coordinate of the starting position of <i>str</i> in the window.</td></tr></table> | <i>str</i> | Is a pointer to the character string that is to be written to the window. | <i>n</i> | Is the maximum number of characters to be copied from <i>str</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of <i>str</i> in the window. |
| <i>str</i> | Is a pointer to the character string that is to be written to the window. | | | | | | |
| <i>n</i> | Is the maximum number of characters to be copied from <i>str</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | | | | | | |
| <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of <i>str</i> in the window. | | | | | | |

addstr(3XCURSES)

x Is the x (column) coordinate of the starting position of *str* in the window.

win Is a pointer to the window in which the string is to be written.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addch(3XCURSES)`, `addchstr(3XCURSES)`, `curses(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

addwch(3CURSES)

| | |
|----------------------|--|
| NAME | <code> curs_addwch, addwch, waddwch, mvaddwch, mvwaddwch, echowchar, wechowchar </code> – add a <code>wchar_t</code> character (with attributes) to a curses window and advance cursor |
| SYNOPSIS | <pre>cc [flag...] file... -lcurses [library...] #include<curses.h> int addwch(chtype <i>wch</i>); int waddwch(WINDOW *<i>win</i>, chtype <i>wch</i>); int mvaddwch(int <i>y</i>, int <i>x</i>, chtype <i>wch</i>); int mvwaddwch(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>, chtype <i>wch</i>); int echowchar(chtype <i>wch</i>); int wechowchar(WINDOW *<i>win</i>, chtype <i>wch</i>);</pre> |
| DESCRIPTION | <p>The <code>addwch()</code>, <code>waddwch()</code>, <code>mvaddwch()</code>, and <code>mvwaddwch()</code> routines put the character <code>wch</code>, holding a <code>wchar_t</code> character, into the window at the current cursor position of the window and advance the position of the window cursor. Their function is similar to that of <code>putwchar(3C)</code> in the C multibyte library. At the right margin, an automatic newline is performed. At the bottom of the scrolling region, if <code>scrollok</code> is enabled, the scrolling region is scrolled up one line.</p> <p>If <code>wch</code> is a tab, newline, or backspace, the cursor is moved appropriately within the window. A newline also does a <code>clrtoeol(3CURSES)</code> before moving. Tabs are considered to be at every eighth column. If <code>wch</code> is another control character, it is drawn in the <code>^X</code> notation. Calling <code>winwch(3CURSES)</code> after adding a control character does not return the control character, but instead returns the representation of the control character.</p> <p>Video attributes can be combined with a <code>wchar_t</code> character by OR-ing them into the parameter. This results in these attributes also being set. (The intent here is that text, including attributes, can be copied from one place to another using <code>inwch()</code> and <code>addwch()</code>.) See <code>standout(3CURSES)</code>, predefined video attribute constants.</p> <p>The <code>echowchar()</code> and <code>wechowchar()</code> routines are functionally equivalent to a call to <code>addwch()</code> followed by a call to <code>refresh(3CURSES)</code>, or a call to <code>waddwch()</code> followed by a call to <code>wrefresh(3CURSES)</code>. The knowledge that only a single character is being output is taken into consideration and, for non-control characters, a considerable performance gain might be seen by using these routines instead of their equivalents.</p> |
| Line Graphics | The following variables may be used to add line drawing characters to the screen with routines of the <code>addwch()</code> family. When variables are defined for the terminal, the <code>A_ALTCHARSET</code> bit is turned on. (See <code> curs_attr(3CURSES)</code>). Otherwise, the default character listed below is stored in the variable. The names chosen are consistent with the VT100 nomenclature. |

addwch(3CURSES)

| Name | Default | Glyph Description |
|--------------|---------|-------------------------|
| ACS_ULCORNER | + | upper left-hand corner |
| ACS_LLCORNER | + | lower left-hand corner |
| ACS_URCORNER | + | upper right-hand corner |
| ACS_LRCORNER | + | lower right-hand corner |
| ACS_RTEE | + | right tee |
| ACS_LTEE | + | left tee |
| ACS_BTEE | + | bottom tee |
| ACS_TTEE | + | top tee |
| ACS_HLINE | - | horizontal line |
| ACS_VLINE | | vertical line |
| ACS_PLUS | + | plus |
| ACS_S1 | - | scan line 1 |
| ACS_S9 | - | scan line 9 |
| ACS_DIAMOND | + | diamond |
| ACS_CKBOARD | : | checker board (stipple) |
| ACS_DEGREE | ' | degree symbol |
| ACS_PLMINUS | # | plus/minus |
| ACS_BULLET | o | bullet |
| ACS_LARROW | < | arrow pointing left |
| ACS_RARRROW | > | arrow pointing right |
| ACS_DARROW | v | arrow pointing down |
| ACS_UARROW | ^ | arrow pointing up |
| ACS_BOARD | # | board of squares |
| ACS_LANTERN | # | lantern symbol |
| ACS_BLOCK | # | solid square block |

RETURN VALUE

All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion, unless otherwise noted in the preceding routine descriptions.

addwch(3CURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `putwchar(3C)`, `clrtoeol(3CURSES)`, `curses(3CURSES)`, `curl_attr(3CURSES)`, `curl_inwch(3CURSES)`, `curl_outopts(3CURSES)`, `refresh(3CURSES)`, `standout(3CURSES)`, `winch(3CURSES)`, `wrefresh(3CURSES)`, `attributes(5)`

NOTES The header file `<curses.h>` automatically includes the header files `<stdio.h>`, `<unctrl.h>` and `<widec.h>`.

Note that `addwch()`, `mvaddwch()`, `mvwaddwch()`, and `echowchar()` may be macros.

None of these routines can use the color attribute in `chtype`.

| | |
|--------------------|--|
| NAME | add_wch, mvadd_wch, mvwadd_wch, wadd_wch – add a complex character (with rendition) to a window |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int add_wch(const cchar_t *wch); int wadd_wch(WINDOW *win, const cchar_t *wch); int mvadd_wch(int y, int x, const cchar_t *wch); int mvwadd_wch(WINDOW *win, int y, int x, const cchar_t *wch);</pre> |
| DESCRIPTION | <p>The <code>add_wch()</code> function writes a complex character to the <code>stdscr</code> window at the current cursor position. The <code>mvadd_wch()</code> and <code>mvwadd_wch()</code> functions write the character to the position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters. The <code>mvadd_wch()</code> function writes the character to the <code>stdscr</code> window, while <code>mvwadd_wch()</code> writes the character to the window specified by <code>win</code>. The <code>wadd_wch()</code> function is identical to <code>add_wch()</code>, but writes the character to the window specified by <code>win</code>. These functions advance the cursor after writing the character.</p> <p>If <code>wch</code> is a spacing complex character, X/Open Curses replaces any previous character at the specified location with <code>wch</code> (and its rendition). If <code>wch</code> is a non-spacing complex character, X/Open Curses preserves all existing characters at the specified location and adds the non-spacing characters of <code>wch</code> to the spacing complex character. It ignores the rendition associated with <code>wch</code>.</p> <p>Characters that do not fit on the end of the current line are wrapped to the beginning of the next line unless the current line is the last line of the window and scrolling is disabled. In that situation, X/Open Curses discards characters which extend beyond the end of the line.</p> <p>When <code>wch</code> is a backspace, carriage return, newline, or tab, X/Open Curses moves the cursor appropriately as described in the <code>curses(3XCURSES)</code> man page. Each tab character moves the cursor to the next tab stop. By default, tab stops occur every eight columns. When <code>wch</code> is a control character other than a backspace, carriage return, newline, or tab, it is written using <code>^x</code> notation, where <code>x</code> is a printable character. When X/Open Curses writes <code>wch</code> to the last character position on a line, it automatically generates a newline. When <code>wch</code> is written to the last character position of a scrolling region and <code>scrollok()</code> is enabled, X/Open Curses scrolls the scrolling region up one line (see <code>clearok(3XCURSES)</code>).</p> |
| PARAMETERS | <p><code>wch</code> Is the character/attribute pair (rendition) to be written to the window.</p> <p><code>win</code> Is a pointer to the window in which the character is to be written.</p> <p><code>y</code> Is the <code>y</code> (row) coordinate of the character's position in the window.</p> |

add_wch(3XCURSES)

x Is the *x* (column) coordinate of the character's position in the window.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attr_off(3XCURSES)`, `bkgrndset(3XCURSES)`, `curses(3XCURSES)`, `doupdate(3XCURSES)`, `in_wch(3XCURSES)`, `ins_wch(3XCURSES)`, `libcurses(3XCURSES)`, `n1(3XCURSES)`, `printw(3XCURSES)`, `scrollok(3XCURSES)`, `scl1(3XCURSES)`, `setscrreg(3XCURSES)`, `terminfo(4)`, `attributes(5)`, `standards(5)`

| NAME | curs_addwchstr, addwchstr, addwchnstr, waddwchstr, waddwchnstr, mvaddwchstr, mvaddwchnstr, mvwaddwchstr, mvwaddwchnstr – add string of wchar_t characters (and attributes) to a curses window | | | | |
|---------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lcurses [library...] #include<curses.h> int addwchstr(chtype *wchstr); int addwchnstr(chtype *wchstr, int n); int waddwchstr(WINDOW *win, chtype *wchstr); int waddwchnstr(WINDOW *win, chtype *wchstr, int n); int mvaddwchstr(int y, int x, chtype *wchstr); int mvaddwchnstr(int y, int x, chtype *wchstr, int n); int mvwaddwchstr(WINDOW *win, int y, int x, chtype *wchstr); int mvwaddwchnstr(WINDOW *win, int y, int x, chtype *wchstr, int n);</pre> | | | | |
| DESCRIPTION | <p>All of these routines copy <i>wchstr</i>, which points to a string of wchar_t characters, directly into the window image structure starting at the current cursor position. The four routines with <i>n</i> as the last argument copy at most <i>n</i> elements, but no more than will fit on the line. If <i>n</i>==–1 then the whole string is copied, to the maximum number that fit on the line.</p> <p>The position of the window cursor is not advanced. These routines work faster than waddnwstr(3CURSES) because they merely copy <i>wchstr</i> into the window image structure. On the other hand, care must be taken when using these functions because they don't perform any kind of checking (such as for the newline character), they do not advance the current cursor position, and they truncate the string, rather than wrapping it around to the new line.</p> | | | | |
| RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion, unless otherwise noted in the preceding routine descriptions. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), waddnwstr(3CURSES), attributes(5) | | | | |
| NOTES | The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <widec.h>. | | | | |

`addwchnstr(3CURSES)`

Note that all routines except `waddwchnstr()` may be macros.

None of these routines can use the color attribute in `chtype`.

| | | | | | | | |
|--------------------|--|---------------|---|----------|--|----------|---|
| NAME | add_wchnstr, add_wchstr, mvadd_wchnstr, mvadd_wchstr, mvwadd_wchnstr, mvwadd_wchstr, wadd_wchnstr, wadd_wchstr – copy a string of complex characters (with renditions) to a window | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int add_wchnstr(const cchar_t *wchstr, int n); int add_wchstr(const cchar_t *wchstr); int mvadd_wchnstr(int y, int x, const cchar_t *wchstr, int n); int mvadd_wchstr(int y, int x, const cchar_t *wchstr); int mvwadd_wchnstr(WINDOW *win, int y, int x, const cchar_t *wchstr, int n); int mvwadd_wchstr(WINDOW *win, int y, int x, const cchar_t *wchstr); int wadd_wchstr(WINDOW *win, const cchar_t *wchstr); int wadd_wchnstr(WINDOW *win, const cchar_t *wchstr, int n);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>add_wchstr()</code> function copies the string of <code>cchar_t</code> characters to the <code>stdscr</code> window at the current cursor position. The <code>mvadd_wchstr()</code> and <code>mvwadd_wchstr()</code> functions copy the string to the starting position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <code>win</code>). The <code>wadd_wchstr()</code> is identical to <code>add_wchstr()</code>, but writes to the window specified by <code>win</code>.</p> <p>The <code>add_wchnstr()</code>, <code>wadd_wchnstr()</code>, <code>mvadd_wchnstr()</code>, and <code>mvwadd_wchnstr()</code> functions write <code>n</code> characters to the window, or as many as will fit on the line. If <code>n</code> is less than 0, the entire string is written, or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <code>x</code> and <code>y</code> parameters.</p> <p>These functions differ from the <code>addwstr(3XCURSES)</code> set of functions in two important respects. First, these functions do <i>not</i> advance the cursor after writing the string to the window. Second, the current window rendition (that is, the combination of attributes and color pair) is not combined with the character; only those attributes that are already part of the <code>cchar_t</code> character are used.</p> | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>wchstr</i></td> <td>Is a pointer to the <code>cchar_t</code> string to be copied to the window.</td> </tr> <tr> <td><i>n</i></td> <td>Is the maximum number of characters to be copied from <i>wchstr</i>. If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line.</td> </tr> <tr> <td><i>y</i></td> <td>Is the <i>y</i> (row) coordinate of the starting position of <i>wchstr</i> in the window.</td> </tr> </table> | <i>wchstr</i> | Is a pointer to the <code>cchar_t</code> string to be copied to the window. | <i>n</i> | Is the maximum number of characters to be copied from <i>wchstr</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of <i>wchstr</i> in the window. |
| <i>wchstr</i> | Is a pointer to the <code>cchar_t</code> string to be copied to the window. | | | | | | |
| <i>n</i> | Is the maximum number of characters to be copied from <i>wchstr</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | | | | | | |
| <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of <i>wchstr</i> in the window. | | | | | | |

add_wchnstr(3XCURSES)

x Is the x (column) coordinate of the starting position of *wchstr* in the window.

win Is a pointer to the window to which the string is to be copied.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addnwstr(3XCURSES)`, `add_wch(3XCURSES)`, `attr_off(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_addwchstr, addwchstr, addwchnstr, waddwchstr, waddwchnstr, mvaddwchstr, mvaddwchnstr, mvwaddwchstr, mvwaddwchnstr – add string of wchar_t characters (and attributes) to a curses window | | | | |
|---------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lcurses [library...] #include<curses.h> int addwchstr(chtype *wchstr); int addwchnstr(chtype *wchstr, int n); int waddwchstr(WINDOW *win, chtype *wchstr); int waddwchnstr(WINDOW *win, chtype *wchstr, int n); int mvaddwchstr(int y, int x, chtype *wchstr); int mvaddwchnstr(int y, int x, chtype *wchstr, int n); int mvwaddwchstr(WINDOW *win, int y, int x, chtype *wchstr); int mvwaddwchnstr(WINDOW *win, int y, int x, chtype *wchstr, int n);</pre> | | | | |
| DESCRIPTION | <p>All of these routines copy <i>wchstr</i>, which points to a string of wchar_t characters, directly into the window image structure starting at the current cursor position. The four routines with <i>n</i> as the last argument copy at most <i>n</i> elements, but no more than will fit on the line. If <i>n</i>==−1 then the whole string is copied, to the maximum number that fit on the line.</p> <p>The position of the window cursor is not advanced. These routines work faster than waddnwstr(3CURSES) because they merely copy <i>wchstr</i> into the window image structure. On the other hand, care must be taken when using these functions because they don't perform any kind of checking (such as for the newline character), they do not advance the current cursor position, and they truncate the string, rather than wrapping it around to the new line.</p> | | | | |
| RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion, unless otherwise noted in the preceding routine descriptions. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), waddnwstr(3CURSES), attributes(5) | | | | |
| NOTES | The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <widec.h>. | | | | |

`addwchstr(3CURSES)`

Note that all routines except `waddwchnstr()` may be macros.

None of these routines can use the color attribute in `chtype`.

| | | | | | | | |
|--------------------|--|---------------|---|----------|--|----------|---|
| NAME | add_wchnstr, add_wchstr, mvadd_wchnstr, mvadd_wchstr, mvwadd_wchnstr, mvwadd_wchstr, wadd_wchnstr, wadd_wchstr – copy a string of complex characters (with renditions) to a window | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int add_wchnstr(const cchar_t *wchstr, int n); int add_wchstr(const cchar_t *wchstr); int mvadd_wchnstr(int y, int x, const cchar_t *wchstr, int n); int mvadd_wchstr(int y, int x, const cchar_t *wchstr); int mvwadd_wchnstr(WINDOW *win, int y, int x, const cchar_t *wchstr, int n); int mvwadd_wchstr(WINDOW *win, int y, int x, const cchar_t *wchstr); int wadd_wchstr(WINDOW *win, const cchar_t *wchstr); int wadd_wchnstr(WINDOW *win, const cchar_t *wchstr, int n);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>add_wchstr()</code> function copies the string of <code>cchar_t</code> characters to the <code>stdscr</code> window at the current cursor position. The <code>mvadd_wchstr()</code> and <code>mvwadd_wchstr()</code> functions copy the string to the starting position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <code>win</code>). The <code>wadd_wchstr()</code> is identical to <code>add_wchstr()</code>, but writes to the window specified by <code>win</code>.</p> <p>The <code>add_wchnstr()</code>, <code>wadd_wchnstr()</code>, <code>mvadd_wchnstr()</code>, and <code>mvwadd_wchnstr()</code> functions write <code>n</code> characters to the window, or as many as will fit on the line. If <code>n</code> is less than 0, the entire string is written, or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <code>x</code> and <code>y</code> parameters.</p> <p>These functions differ from the <code>addwstr(3XCURSES)</code> set of functions in two important respects. First, these functions do <i>not</i> advance the cursor after writing the string to the window. Second, the current window rendition (that is, the combination of attributes and color pair) is not combined with the character; only those attributes that are already part of the <code>cchar_t</code> character are used.</p> | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>wchstr</i></td> <td>Is a pointer to the <code>cchar_t</code> string to be copied to the window.</td> </tr> <tr> <td><i>n</i></td> <td>Is the maximum number of characters to be copied from <i>wchstr</i>. If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line.</td> </tr> <tr> <td><i>y</i></td> <td>Is the <i>y</i> (row) coordinate of the starting position of <i>wchstr</i> in the window.</td> </tr> </table> | <i>wchstr</i> | Is a pointer to the <code>cchar_t</code> string to be copied to the window. | <i>n</i> | Is the maximum number of characters to be copied from <i>wchstr</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of <i>wchstr</i> in the window. |
| <i>wchstr</i> | Is a pointer to the <code>cchar_t</code> string to be copied to the window. | | | | | | |
| <i>n</i> | Is the maximum number of characters to be copied from <i>wchstr</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | | | | | | |
| <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of <i>wchstr</i> in the window. | | | | | | |

add_wchstr(3XCURSES)

x Is the x (column) coordinate of the starting position of *wchstr* in the window.

win Is a pointer to the window to which the string is to be copied.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addnwstr(3XCURSES)`, `add_wch(3XCURSES)`, `attr_off(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_addwstr, addwstr, addnwstr, waddwstr, waddnwstr, mvaddwstr, mvaddnwstr, mvwaddwstr, mvwaddnwstr – add a string of wchar_t characters to a curses window and advance cursor | | | | |
|---------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lcurses [library...] #include<curses.h> int addwstr(wchar_t *wstr); int addnwstr(wchar_t *wstr, int n); int waddwstr(WINDOW *win, wchar_t *wstr); int waddnwstr(WINDOW *win, wchar_t *wstr, int n); int mvaddwstr(int y, int x, wchar_t *wstr); int mvaddnwstr(int y, int x, wchar_t *wstr, int n); int mvwaddwstr(WINDOW *win, int y, int x, wchar_t *wstr); int mvwaddnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);</pre> | | | | |
| DESCRIPTION | All of these routines write all the characters of the null-terminated wchar_t character string wstr on the given window. The effect is similar to calling waddwch(3CURSES) once for each wchar_t character in the string. The four routines with n as the last argument write at most n wchar_t characters. If n is negative, then the entire string will be added. | | | | |
| RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), waddwch(3CURSES), attributes(5) | | | | |
| NOTES | <p>The header file <curses.h> automatically includes the header files <stdio.h>, <nctrl.h> and <widec.h>.</p> <p>Note that all of these routines except waddwstr() and waddnwstr() may be macros.</p> | | | | |

addwstr(3XCURSES)

| | |
|--------------------|---|
| NAME | <code>addnwstr</code> , <code>addwstr</code> , <code>mvaddnwstr</code> , <code>mvaddwstr</code> , <code>mvwaddnwstr</code> , <code>mvwaddwstr</code> , <code>waddnwstr</code> , <code>waddwstr</code> – add a wide-character string to a window |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addnwstr(const wchar_t *wstr, int n); int addwstr(const wchar_t *wstr); int mvaddnwstr(int y, int x, const wchar_t *wstr, int n); int mvaddwstr(int y, int x, const wchar_t *wstr); int mvwaddnwstr(WINDOW*win, int y, int x, const wchar_t *wstr, int n); int mvwaddwstr(WINDOW*win, int y, int x, const wchar_t *wstr); int waddnwstr(WINDOW*win, const wchar_t *wstr, int n); int waddwstr(WINDOW*win, const wchar_t *wstr);</pre> |
| DESCRIPTION | <p>The <code>addwstr()</code> function writes a null-terminated wide-character string to the <code>stdscr</code> window at the current cursor position. The <code>waddwstr()</code> function performs an identical action, but writes the string to the window specified by <code>win</code>. The <code>mvaddwstr()</code> and <code>mvwaddwstr()</code> functions write the string to the position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <code>win</code>).</p> <p>The <code>addnwstr()</code>, <code>waddnwstr()</code>, <code>mvaddnwstr()</code>, and <code>mvwaddnwstr()</code> functions write at most <code>n</code> characters to the window. If <code>n</code> is less than 0, the entire string is written. The former two functions place the characters at the current cursor position; the latter two commands use the position specified by the <code>x</code> and <code>y</code> parameters.</p> <p>All of these functions advance the cursor after writing the string.</p> <p>These functions are functionally equivalent to building a <code>cchar_t</code> from the <code>wchar_t</code> and the window rendition (or background character and rendition) and calling the <code>wadd_wch(3XCURSES)</code> function once for each <code>wchar_t</code> in the string. Refer to the <code>curses(3XCURSES)</code> man page for a complete description of special character handling and of the interaction between the window rendition (or background character and rendition) and the character written.</p> <p>Note that these functions differ from the <code>add_wchnstr(3XCURSES)</code> set of functions in that the latter copy the string as is (without combining each character with the foreground and background attributes of the window).</p> |
| PARAMETERS | <p><code>wstr</code> Is a pointer to the wide-character string that is to be written to the window.</p> |

addwstr(3XCURSES)

- n* Is the maximum number of characters to be copied from *wstr*. If *n* is less than 0, the entire string is written or as much of it as fits on the line.
- y* Is the y (row) coordinate of the starting position of *wstr* in the window.
- x* Is the x (column) coordinate of the starting position of *wstr* in the window.
- win* Is a pointer to the window in which the string is to be written.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `add_wch(3XCURSES)`, `add_wchnstr(3XCURSES)`, `curses(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

adjcurspos(3CURSES)

| NAME | <code>curs_alecompat</code> , <code>movenextch</code> , <code>wmovenextch</code> , <code>moveprevch</code> , <code>wmoveprevch</code> , <code>adjcurspos</code> , <code>wadjcurspos</code> – these functions are added to ALE curses library for moving the cursor by character. | | | | |
|---------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int movenextch(void); int wmovenextch(WINDOW *win); int moveprevch(void); int wmoveprevch(WINDOW *win); int adjcurspos(void); int wadjcurspos(WINDOW *win);</pre> | | | | |
| DESCRIPTION | <p><code>movenextch()</code> and <code>wmovenextch()</code> move the cursor to the next character to the right. If the next character is a multicolumn character, the cursor is positioned on the first (left-most) column of that character. The new cursor position will be on the next character, even if the cursor was originally positioned on the left-most column of a multicolumn character. Note that the simple cursor increment (<code>++x</code>) does not guarantee movement to the next character, if the cursor was originally positioned on a multicolumn character. <code>getyx(3CURSES)</code> can be used to find the new position.</p> <p><code>moveprevch()</code> and <code>wmoveprevch()</code> routines are the opposite of <code>movenextch()</code> and <code>wmovenextch()</code>, moving the cursor to the left-most column of the previous character.</p> <p><code>adjcurspos()</code> and <code>wadjcurspos()</code> move the cursor to the first(left-most) column of the multicolumn character that the cursor is presently on. If the cursor is already on the first column, or if the cursor is on a single-column character, these routines will have no effect.</p> | | | | |
| RETURN VALUE | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>getyx(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header file <code><curses.h></code> automatically includes the header files <code><stdio.h></code> , <code><unctrl.h></code> and <code><widec.h></code> . Note that <code>movenextch()</code> , <code>moveprevch()</code> , and <code>adjcurspos()</code> may be macros. | | | | |

| | |
|--------------------|--|
| NAME | plot, arc, box, circle, closepl, closevt, cont, erase, label, line, linemod, move, openpl, openvt, point, space – graphics interface |
| SYNOPSIS | <pre>cc [flag ...] file ... -lplot [library...] #include <plot.h> void arc(short x0, short y0, short x1, short y1, short x2, short y2); void box(short x0, short y0, short x1, short y1); void circle(short x, short y, short r); void closepl(); void closevt(); void cont(short x, short y); void erase(); void label(char *s); void line(short x0, short y0, short x1, short y1); void linemod(char *s); void move(short x, short y); void openpl(); void openvt(); void point(short x, short y); void space(short x0, short y0, short x1, short y1);</pre> |
| DESCRIPTION | <p>These functions generate graphics output for a set of output devices. The format of the output is dependent upon which link editor option is used when the program is compiled and linked (see Link Editor).</p> <p>The term "current point" refers to the current setting for the x and y coordinates.</p> <p>The <code>arc()</code> function specifies a circular arc. The coordinates $(x0, y0)$ specify the center of the arc. The coordinates $(x1, y1)$ specify the starting point of the arc. The coordinates $(x2, y2)$ specify the end point of the circular arc.</p> <p>The <code>box()</code> function specifies a rectangle with coordinates $(x0, y0)$, $(x0, y1)$, $(x1, y0)$, and $(x1, y1)$. The current point is set to $(x1, y1)$.</p> <p>The <code>circle()</code> function specifies a circle with a center at the coordinates (x, y) and a radius of r.</p> <p>The <code>closevt()</code> and <code>closepl()</code> functions flush the output.</p> <p>The <code>cont()</code> function specifies a line beginning at the current point and ending at the coordinates (x, y). The current point is set to (x, y).</p> |

arc(3PLOT)

The `erase()` function starts another frame of output.

The `label()` function places the null terminated string `s` so that the first character falls on the current point. The string is then terminated by a NEWLINE character.

The `line()` function draws a line starting at the coordinates $(x0, y0)$ and ending at the coordinates $(x1, y1)$. The current point is set to $(x1, y1)$.

The `linemod()` function specifies the style for drawing future lines. `s` may contain one of the following: `dotted`, `solid`, `longdashed`, `shortdashed`, or `dotdashed`.

The `move()` function sets the current point to the coordinates (x, y) .

The `openpl()` or `openvt()` function must be called to open the device before any other plot functions are called.

The `point()` function plots the point given by the coordinates (x, y) . The current point is set to (x, y) .

The `space()` function specifies the size of the plotting area. The plot will be reduced or enlarged as necessary to fit the area specified. The coordinates $(x0, y0)$ specify the lower left hand corner of the plotting area. The coordinates $(x1, y1)$ specify the upper right hand corner of the plotting area.

Link Editor Various flavors of these functions exist for different output devices. They are obtained by using the following `ld(1)` options:

| | |
|---------------------|--|
| <code>-lplot</code> | device-independent graphics stream on standard output in the format described in <code>plot(4B)</code> |
| <code>-l300</code> | GSI 300 terminal |
| <code>-l300s</code> | GSI 300S terminal |
| <code>-l4014</code> | Tektronix 4014 terminal |
| <code>-l450</code> | GSI 450 terminal |
| <code>-lvt0</code> | |

| | |
|--------------|--|
| FILES | <code>/usr/lib/libplot.a</code> archive library |
| | <code>/usr/lib/libplot.so.1</code> shared object |
| | <code>/usr/lib/sparcv9/libplot.so.1</code> 64-bit shared object |
| | <code>/usr/lib/lib300.a</code> archive library |
| | <code>/usr/lib/lib300.so.1</code> shared object |

```

/usr/lib/sparcv9/lib300.so.1
  64-bit shared object
/usr/lib/lib300s.a
  archive library
/usr/lib/lib300s.so.1
  shared object
/usr/lib/sparcv9/lib300s.so.1
  64-bit shared object
/usr/lib/lib4014.a
  archive library
/usr/lib/lib4014.so.1
  shared object
/usr/lib/sparcv9/lib4014.so.1
  64-bit shared object
/usr/lib/lib450.a
  archive library
/usr/lib/lib450.so.1
  shared object
/usr/lib/sparcv9/lib450.so.1
  64-bit shared object
/usr/lib/libvt0.a
  archive library
/usr/lib/libvt0.so.1
  shared object
/usr/lib/sparcv9/libvt0.so.1
  64-bit shared object

```

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `graph(1)`, `ld(1)`, `libplot(3LIB)`, `plot(4B)`, `attributes(5)`

attr_get(3XCURSES)

| | | | | | | | | | |
|--------------------|--|--------------|--|--------------|---------------------------------------|-------------|-----------------------------|------------|---|
| NAME | attr_get, attr_off, attr_on, attr_set, color_set, wattr_get, wattr_off, wattr_on, wattr_set, wcolor_set – control window attributes | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int attr_get(attr_t *attrs, short *color, void *opts); int attr_off(attr_t attrs, void *opts); int attr_on(attr_t attrs, void *opts); int attr_set(attr_t attrs, short color, void *opts); int color_set(short *color, void *opts); int wattr_get(WINDOW *win, attr_t attrs, short *color, void *opts); int wattr_off(WINDOW *win, attr_t attrs, void *opts); int wattr_on(WINDOW *win, attr_t attrs, void *opts); int wattr_set(WINDOW *win, attr_t attrs, short color, void *opts); int wcolor_set(WINDOW *win, short color, void *opts);</pre> | | | | | | | | |
| DESCRIPTION | <p>The attr_get() function retrieves the current rendition of <i>stdscr</i>. The wattr_get() function retrieves the current rendition of window <i>win</i>. If <i>attrs</i> or <i>color</i> is a null pointer, no information is retrieved.</p> <p>The attr_off() and attr_on() functions unset and set, respectively, the specified window attributes of <i>stdscr</i>. These functions only affect the attributes specified; attributes that existed before the call are retained.</p> <p>The wattr_off() and wattr_on() functions unset or set the specified attributes for window <i>win</i>.</p> <p>The attr_set() and wattr_set() functions change the rendition of <i>stdscr</i> and <i>win</i>; the old values are not retained.</p> <p>The color_set() and wcolor_set() functions set the window color of <i>stdscr</i> and <i>win</i> to <i>color</i>.</p> <p>The attributes and color pairs that can be used are specified in the Attributes, Color Pairs, and Renditions section of the curses(3XCURSES) man page.</p> | | | | | | | | |
| PARAMETERS | <table><tr><td><i>attrs</i></td><td>Is a pointer to the foreground window attributes to be set or unset.</td></tr><tr><td><i>color</i></td><td>Is a pointer to a color pair number .</td></tr><tr><td><i>opts</i></td><td>Is reserved for future use.</td></tr><tr><td><i>win</i></td><td>Is a pointer to the window in which attribute changes are to be made.</td></tr></table> | <i>attrs</i> | Is a pointer to the foreground window attributes to be set or unset. | <i>color</i> | Is a pointer to a color pair number . | <i>opts</i> | Is reserved for future use. | <i>win</i> | Is a pointer to the window in which attribute changes are to be made. |
| <i>attrs</i> | Is a pointer to the foreground window attributes to be set or unset. | | | | | | | | |
| <i>color</i> | Is a pointer to a color pair number . | | | | | | | | |
| <i>opts</i> | Is reserved for future use. | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which attribute changes are to be made. | | | | | | | | |

attr_get(3XCURSES)

RETURN VALUES These functions always return OK.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO add_wch(3XCURSES), addnwstr(3XCURSES), attroff(3XCURSES), bkgrndset(3XCURSES), curses(3XCURSES), init_color(3XCURSES), libcurses(3XCURSES), start_color(3XCURSES), attributes(5), standards(5)

attroff(3CURSES)

| | | | | | | | | | | | | | | | |
|---------------------------|--|-------------------------|--|--------------------------|-------------|------------------------|---------------|----------------------|----------|--------------------|-------------|---------------------|----------------------|---------------------------|-------------------------|
| NAME | <code>curs_attr</code> , <code>attroff</code> , <code>wattroff</code> , <code>attron</code> , <code>wattron</code> , <code>attrset</code> , <code>wattrset</code> , <code>standend</code> , <code>wstandend</code> , <code>standout</code> , <code>wstandout</code> – curses character and window attribute control routines | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int attroff(int attrs); int wattroff(WINDOW *win, int attrs); int attron(int attrs); int wattron(WINDOW *win, int attrs); int attrset(int attrs); int wattrset(WINDOW *win, int attrs); int standend(void); int wstandend(WINDOW *win); int standout(void); int wstandout(WINDOW *win);</pre> | | | | | | | | | | | | | | |
| DESCRIPTION | <p>All of these routines manipulate the current attributes of the named window. The current attributes of a window are applied to all characters that are written into the window with <code>waddch()</code>, <code>waddstr()</code>, and <code>wprintw()</code>. Attributes are a property of the character, and move with the character through any scrolling and insert/delete line/character operations. To the extent possible on the particular terminal, they are displayed as the graphic rendition of characters put on the screen.</p> <p>The routine <code>attrset()</code> sets the current attributes of the given window to <i>attrs</i>. The routine <code>attroff()</code> turns off the named attributes without turning any other attributes on or off. The routine <code>attron()</code> turns on the named attributes without affecting any others. The routine <code>standout()</code> is the same as <code>attron(A_STANDOUT)</code>. The routine <code>standend()</code> is the same as <code>attrset()</code>, that is, it turns off all attributes.</p> | | | | | | | | | | | | | | |
| Attributes | <p>The following video attributes, defined in <code><curses.h></code>, can be passed to the routines <code>attron()</code>, <code>attroff()</code>, and <code>attrset()</code>, or OR-ed with the characters passed to <code>addch()</code>.</p> <table><tr><td><code>A_STANDOUT</code></td><td>Best highlighting mode of the terminal</td></tr><tr><td><code>A_UNDERLINE</code></td><td>Underlining</td></tr><tr><td><code>A_REVERSE</code></td><td>Reverse video</td></tr><tr><td><code>A_BLINK</code></td><td>Blinking</td></tr><tr><td><code>A_DIM</code></td><td>Half bright</td></tr><tr><td><code>A_BOLD</code></td><td>Extra bright or bold</td></tr><tr><td><code>A_ALTCHARSET</code></td><td>Alternate character set</td></tr></table> | <code>A_STANDOUT</code> | Best highlighting mode of the terminal | <code>A_UNDERLINE</code> | Underlining | <code>A_REVERSE</code> | Reverse video | <code>A_BLINK</code> | Blinking | <code>A_DIM</code> | Half bright | <code>A_BOLD</code> | Extra bright or bold | <code>A_ALTCHARSET</code> | Alternate character set |
| <code>A_STANDOUT</code> | Best highlighting mode of the terminal | | | | | | | | | | | | | | |
| <code>A_UNDERLINE</code> | Underlining | | | | | | | | | | | | | | |
| <code>A_REVERSE</code> | Reverse video | | | | | | | | | | | | | | |
| <code>A_BLINK</code> | Blinking | | | | | | | | | | | | | | |
| <code>A_DIM</code> | Half bright | | | | | | | | | | | | | | |
| <code>A_BOLD</code> | Extra bright or bold | | | | | | | | | | | | | | |
| <code>A_ALTCHARSET</code> | Alternate character set | | | | | | | | | | | | | | |

attroff(3CURSES)

A_CHARTEXT Bit-mask to extract a character

COLOR_PAIR(*n*) Color-pair number *n*

The following macro is the reverse of COLOR_PAIR(*n*) :

PAIR_NUMBER(*attrs*) Returns the pair number associated with the COLOR_PAIR(*n*) attribute

RETURN VALUES These routines always return 1.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_addch(3CURSES), curs_addstr(3CURSES), curs_printw(3CURSES), curses(3CURSES), attributes(5)

NOTES The header < curses.h > automatically includes the headers < stdio.h > and < unctrl.h >.

Note that attroff(), wattroff(), attron(), wattron(), wattrset(), standend(), and standout() may be macros.

attr_off(3XCURSES)

| | | | | | | | | | |
|--------------------|--|--------------|--|--------------|---------------------------------------|-------------|-----------------------------|------------|---|
| NAME | attr_get, attr_off, attr_on, attr_set, color_set, wattr_get, wattr_off, wattr_on, wattr_set, wcolor_set – control window attributes | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int attr_get(attr_t *attrs, short *color, void *opts); int attr_off(attr_t attrs, void *opts); int attr_on(attr_t attrs, void *opts); int attr_set(attr_t attrs, short color, void *opts); int color_set(short *color, void *opts); int wattr_get(WINDOW *win, attr_t attrs, short *color, void *opts); int wattr_off(WINDOW *win, attr_t attrs, void *opts); int wattr_on(WINDOW *win, attr_t attrs, void *opts); int wattr_set(WINDOW *win, attr_t attrs, short color, void *opts); int wcolor_set(WINDOW *win, short color, void *opts);</pre> | | | | | | | | |
| DESCRIPTION | <p>The attr_get() function retrieves the current rendition of <i>stdscr</i>. The wattr_get() function retrieves the current rendition of window <i>win</i>. If <i>attrs</i> or <i>color</i> is a null pointer, no information is retrieved.</p> <p>The attr_off() and attr_on() functions unset and set, respectively, the specified window attributes of <i>stdscr</i>. These functions only affect the attributes specified; attributes that existed before the call are retained.</p> <p>The wattr_off() and wattr_on() functions unset or set the specified attributes for window <i>win</i>.</p> <p>The attr_set() and wattr_set() functions change the rendition of <i>stdscr</i> and <i>win</i>; the old values are not retained.</p> <p>The color_set() and wcolor_set() functions set the window color of <i>stdscr</i> and <i>win</i> to <i>color</i>.</p> <p>The attributes and color pairs that can be used are specified in the Attributes, Color Pairs, and Renditions section of the curses(3XCURSES) man page.</p> | | | | | | | | |
| PARAMETERS | <table><tr><td><i>attrs</i></td><td>Is a pointer to the foreground window attributes to be set or unset.</td></tr><tr><td><i>color</i></td><td>Is a pointer to a color pair number .</td></tr><tr><td><i>opts</i></td><td>Is reserved for future use.</td></tr><tr><td><i>win</i></td><td>Is a pointer to the window in which attribute changes are to be made.</td></tr></table> | <i>attrs</i> | Is a pointer to the foreground window attributes to be set or unset. | <i>color</i> | Is a pointer to a color pair number . | <i>opts</i> | Is reserved for future use. | <i>win</i> | Is a pointer to the window in which attribute changes are to be made. |
| <i>attrs</i> | Is a pointer to the foreground window attributes to be set or unset. | | | | | | | | |
| <i>color</i> | Is a pointer to a color pair number . | | | | | | | | |
| <i>opts</i> | Is reserved for future use. | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which attribute changes are to be made. | | | | | | | | |

attr_off(3XCURSES)

RETURN VALUES These functions always return OK.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO add_wch(3XCURSES), addnwstr(3XCURSES), attroff(3XCURSES), bkgrndset(3XCURSES), curses(3XCURSES), init_color(3XCURSES), libcurses(3XCURSES), start_color(3XCURSES), attributes(5), standards(5)

attroff(3XCURSES)

| | |
|----------------------|--|
| NAME | attroff, attron, attrset, wattroff, wattron, wattrset – change foreground window attributes |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int attroff(int attrs); int attron(int attrs); int attrset(int attrs); int wattroff(WINDOW *win, int attrs); int wattron(WINDOW *win, int attrs); int wattrset(WINDOW *win, int attrs);</pre> |
| DESCRIPTION | <p>The <code>attroff()</code> and <code>attron()</code> functions unset and set, respectively, the specified window attributes of <code>stdscr</code>. These functions only affect the attributes specified; attributes that existed before the call are retained. The <code>wattroff()</code> and <code>wattron()</code> functions unset or set the specified attributes for window <i>win</i>.</p> <p>The <code>attrset()</code> and <code>wattrset()</code> functions change the specified window renditions of <code>stdscr</code> and <i>win</i> to new values; the old values are not retained.</p> <p>The attributes that can be used are specified in the <i>Attributes, Color Pairs, and Renditions</i> section of the <code>curses(3XCURSES)</code> man page.</p> <p>Here is an example that prints some text using the current window rendition, adds underlining, changes the attributes, prints more text, then changes the attributes back.</p> <pre>printw("This word is"); attron(A_UNDERLINE); printw("underlined."); attroff(A_NORMAL); printw("This is back to normal text.\n"); refresh();</pre> |
| PARAMETERS | <p><i>attrs</i> are the foreground window attributes to be set or unset.</p> <p><i>win</i> Is a pointer to the window in which attribute changes are to be made.</p> |
| RETURN VALUES | These functions always return OK or 1. |
| ERRORS | None. |
| USAGE | All of these functions may be macros. |

attroff(3XCURSES)

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO addch(3XCURSES), addnstr(3XCURSES), attr_get(3XCURSES), bkgdset(3XCURSES), curses(3XCURSES), init_color(3XCURSES), libcurses(3XCURSES), start_color(3XCURSES), attributes(5), standards(5)

attron(3CURSES)

| | | | | | | | | | | | | | | | |
|---------------------------|--|-------------------------|--|--------------------------|-------------|------------------------|---------------|----------------------|----------|--------------------|-------------|---------------------|----------------------|---------------------------|-------------------------|
| NAME | <code> curs_attr, attroff, wattroff, attron, wattron, attrset, wattrset, standend, wstandend, standout, wstandout</code> – curses character and window attribute control routines | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int attroff(int attrs); int wattroff(WINDOW *win, int attrs); int attron(int attrs); int wattron(WINDOW *win, int attrs); int attrset(int attrs); int wattrset(WINDOW *win, int attrs); int standend(void); int wstandend(WINDOW *win); int standout(void); int wstandout(WINDOW *win);</pre> | | | | | | | | | | | | | | |
| DESCRIPTION | <p>All of these routines manipulate the current attributes of the named window. The current attributes of a window are applied to all characters that are written into the window with <code>waddch()</code>, <code>waddstr()</code>, and <code>wprintw()</code>. Attributes are a property of the character, and move with the character through any scrolling and insert/delete line/character operations. To the extent possible on the particular terminal, they are displayed as the graphic rendition of characters put on the screen.</p> <p>The routine <code>attrset()</code> sets the current attributes of the given window to <code>attrs</code>. The routine <code>attroff()</code> turns off the named attributes without turning any other attributes on or off. The routine <code>attron()</code> turns on the named attributes without affecting any others. The routine <code>standout()</code> is the same as <code>attron(A_STANDOUT)</code>. The routine <code>standend()</code> is the same as <code>attroff()</code>, that is, it turns off all attributes.</p> | | | | | | | | | | | | | | |
| Attributes | <p>The following video attributes, defined in <code><curses.h></code>, can be passed to the routines <code>attron()</code>, <code>attroff()</code>, and <code>attrset()</code>, or OR-ed with the characters passed to <code>addch()</code>.</p> <table><tr><td><code>A_STANDOUT</code></td><td>Best highlighting mode of the terminal</td></tr><tr><td><code>A_UNDERLINE</code></td><td>Underlining</td></tr><tr><td><code>A_REVERSE</code></td><td>Reverse video</td></tr><tr><td><code>A_BLINK</code></td><td>Blinking</td></tr><tr><td><code>A_DIM</code></td><td>Half bright</td></tr><tr><td><code>A_BOLD</code></td><td>Extra bright or bold</td></tr><tr><td><code>A_ALTCHARSET</code></td><td>Alternate character set</td></tr></table> | <code>A_STANDOUT</code> | Best highlighting mode of the terminal | <code>A_UNDERLINE</code> | Underlining | <code>A_REVERSE</code> | Reverse video | <code>A_BLINK</code> | Blinking | <code>A_DIM</code> | Half bright | <code>A_BOLD</code> | Extra bright or bold | <code>A_ALTCHARSET</code> | Alternate character set |
| <code>A_STANDOUT</code> | Best highlighting mode of the terminal | | | | | | | | | | | | | | |
| <code>A_UNDERLINE</code> | Underlining | | | | | | | | | | | | | | |
| <code>A_REVERSE</code> | Reverse video | | | | | | | | | | | | | | |
| <code>A_BLINK</code> | Blinking | | | | | | | | | | | | | | |
| <code>A_DIM</code> | Half bright | | | | | | | | | | | | | | |
| <code>A_BOLD</code> | Extra bright or bold | | | | | | | | | | | | | | |
| <code>A_ALTCHARSET</code> | Alternate character set | | | | | | | | | | | | | | |

attron(3CURSES)

A_CHARTEXT Bit-mask to extract a character

COLOR_PAIR(*n*) Color-pair number *n*

The following macro is the reverse of COLOR_PAIR(*n*) :

PAIR_NUMBER(*attrs*) Returns the pair number associated with the COLOR_PAIR(*n*) attribute

RETURN VALUES These routines always return 1.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_addch(3CURSES), curs_addstr(3CURSES), curs_printw(3CURSES), curses(3CURSES), attributes(5)

NOTES The header < curses.h > automatically includes the headers < stdio.h > and < unctrl.h >.

Note that attroff(), wattroff(), attron(), wattron(), wattrset(), standend(), and standout() may be macros.

attr_on(3XCURSES)

| | | | | | | | | | |
|--------------------|--|--------------|--|--------------|---------------------------------------|-------------|-----------------------------|------------|---|
| NAME | attr_get, attr_off, attr_on, attr_set, color_set, wattr_get, wattr_off, wattr_on, wattr_set, wcolor_set – control window attributes | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int attr_get(attr_t *attrs, short *color, void *opts); int attr_off(attr_t attrs, void *opts); int attr_on(attr_t attrs, void *opts); int attr_set(attr_t attrs, short color, void *opts); int color_set(short *color, void *opts); int wattr_get(WINDOW *win, attr_t attrs, short *color, void *opts); int wattr_off(WINDOW *win, attr_t attrs, void *opts); int wattr_on(WINDOW *win, attr_t attrs, void *opts); int wattr_set(WINDOW *win, attr_t attrs, short color, void *opts); int wcolor_set(WINDOW *win, short color, void *opts);</pre> | | | | | | | | |
| DESCRIPTION | <p>The attr_get() function retrieves the current rendition of <i>stdscr</i>. The wattr_get() function retrieves the current rendition of window <i>win</i>. If <i>attrs</i> or <i>color</i> is a null pointer, no information is retrieved.</p> <p>The attr_off() and attr_on() functions unset and set, respectively, the specified window attributes of <i>stdscr</i>. These functions only affect the attributes specified; attributes that existed before the call are retained.</p> <p>The wattr_off() and wattr_on() functions unset or set the specified attributes for window <i>win</i>.</p> <p>The attr_set() and wattr_set() functions change the rendition of <i>stdscr</i> and <i>win</i>; the old values are not retained.</p> <p>The color_set() and wcolor_set() functions set the window color of <i>stdscr</i> and <i>win</i> to <i>color</i>.</p> <p>The attributes and color pairs that can be used are specified in the Attributes, Color Pairs, and Renditions section of the curses(3XCURSES) man page.</p> | | | | | | | | |
| PARAMETERS | <table><tr><td><i>attrs</i></td><td>Is a pointer to the foreground window attributes to be set or unset.</td></tr><tr><td><i>color</i></td><td>Is a pointer to a color pair number .</td></tr><tr><td><i>opts</i></td><td>Is reserved for future use.</td></tr><tr><td><i>win</i></td><td>Is a pointer to the window in which attribute changes are to be made.</td></tr></table> | <i>attrs</i> | Is a pointer to the foreground window attributes to be set or unset. | <i>color</i> | Is a pointer to a color pair number . | <i>opts</i> | Is reserved for future use. | <i>win</i> | Is a pointer to the window in which attribute changes are to be made. |
| <i>attrs</i> | Is a pointer to the foreground window attributes to be set or unset. | | | | | | | | |
| <i>color</i> | Is a pointer to a color pair number . | | | | | | | | |
| <i>opts</i> | Is reserved for future use. | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which attribute changes are to be made. | | | | | | | | |

attr_on(3XCURSES)

RETURN VALUES These functions always return OK.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO add_wch(3XCURSES), addnwstr(3XCURSES), attroff(3XCURSES), bkgrndset(3XCURSES), curses(3XCURSES), init_color(3XCURSES), libcurses(3XCURSES), start_color(3XCURSES), attributes(5), standards(5)

attron(3XCURSES)

| | |
|----------------------|--|
| NAME | attroff, attron, attrset, wattroff, wattron, wattrset – change foreground window attributes |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int attroff(int attrs); int attron(int attrs); int attrset(int attrs); int wattroff(WINDOW *win, int attrs); int wattron(WINDOW *win, int attrs); int wattrset(WINDOW *win, int attrs);</pre> |
| DESCRIPTION | <p>The <code>attroff()</code> and <code>attron()</code> functions unset and set, respectively, the specified window attributes of <code>stdscr</code>. These functions only affect the attributes specified; attributes that existed before the call are retained. The <code>wattroff()</code> and <code>wattron()</code> functions unset or set the specified attributes for window <i>win</i>.</p> <p>The <code>attrset()</code> and <code>wattrset()</code> functions change the specified window renditions of <code>stdscr</code> and <i>win</i> to new values; the old values are not retained.</p> <p>The attributes that can be used are specified in the <i>Attributes, Color Pairs, and Renditions</i> section of the <code>curses(3XCURSES)</code> man page.</p> <p>Here is an example that prints some text using the current window rendition, adds underlining, changes the attributes, prints more text, then changes the attributes back.</p> <pre>printw("This word is"); attron(A_UNDERLINE); printw("underlined."); attroff(A_NORMAL); printw("This is back to normal text.\n"); refresh();</pre> |
| PARAMETERS | <p><i>attrs</i> are the foreground window attributes to be set or unset.</p> <p><i>win</i> Is a pointer to the window in which attribute changes are to be made.</p> |
| RETURN VALUES | These functions always return OK or 1. |
| ERRORS | None. |
| USAGE | All of these functions may be macros. |

attron(3XCURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addch(3XCURSES)`, `addnstr(3XCURSES)`, `attr_get(3XCURSES)`, `bkgdset(3XCURSES)`, `curses(3XCURSES)`, `init_color(3XCURSES)`, `libcurses(3XCURSES)`, `start_color(3XCURSES)`, `attributes(5)`, `standards(5)`

attrset(3CURSES)

| | | | | | | | | | | | | | | | |
|---------------------------|--|-------------------------|--|--------------------------|-------------|------------------------|---------------|----------------------|----------|--------------------|-------------|---------------------|----------------------|---------------------------|-------------------------|
| NAME | <code>curs_attr</code> , <code>attroff</code> , <code>wattroff</code> , <code>attron</code> , <code>wattron</code> , <code>attrset</code> , <code>wattrset</code> , <code>standend</code> , <code>wstandend</code> , <code>standout</code> , <code>wstandout</code> – curses character and window attribute control routines | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int attroff(int attrs); int wattroff(WINDOW *win, int attrs); int attron(int attrs); int wattron(WINDOW *win, int attrs); int attrset(int attrs); int wattrset(WINDOW *win, int attrs); int standend(void); int wstandend(WINDOW *win); int standout(void); int wstandout(WINDOW *win);</pre> | | | | | | | | | | | | | | |
| DESCRIPTION | <p>All of these routines manipulate the current attributes of the named window. The current attributes of a window are applied to all characters that are written into the window with <code>waddch()</code>, <code>waddstr()</code>, and <code>wprintw()</code>. Attributes are a property of the character, and move with the character through any scrolling and insert/delete line/character operations. To the extent possible on the particular terminal, they are displayed as the graphic rendition of characters put on the screen.</p> <p>The routine <code>attrset()</code> sets the current attributes of the given window to <i>attrs</i>. The routine <code>attroff()</code> turns off the named attributes without turning any other attributes on or off. The routine <code>attron()</code> turns on the named attributes without affecting any others. The routine <code>standout()</code> is the same as <code>attron(A_STANDOUT)</code>. The routine <code>standend()</code> is the same as <code>attroff()</code>, that is, it turns off all attributes.</p> | | | | | | | | | | | | | | |
| Attributes | <p>The following video attributes, defined in <code><curses.h></code>, can be passed to the routines <code>attron()</code>, <code>attroff()</code>, and <code>attrset()</code>, or OR-ed with the characters passed to <code>addch()</code>.</p> <table><tr><td><code>A_STANDOUT</code></td><td>Best highlighting mode of the terminal</td></tr><tr><td><code>A_UNDERLINE</code></td><td>Underlining</td></tr><tr><td><code>A_REVERSE</code></td><td>Reverse video</td></tr><tr><td><code>A_BLINK</code></td><td>Blinking</td></tr><tr><td><code>A_DIM</code></td><td>Half bright</td></tr><tr><td><code>A_BOLD</code></td><td>Extra bright or bold</td></tr><tr><td><code>A_ALTCHARSET</code></td><td>Alternate character set</td></tr></table> | <code>A_STANDOUT</code> | Best highlighting mode of the terminal | <code>A_UNDERLINE</code> | Underlining | <code>A_REVERSE</code> | Reverse video | <code>A_BLINK</code> | Blinking | <code>A_DIM</code> | Half bright | <code>A_BOLD</code> | Extra bright or bold | <code>A_ALTCHARSET</code> | Alternate character set |
| <code>A_STANDOUT</code> | Best highlighting mode of the terminal | | | | | | | | | | | | | | |
| <code>A_UNDERLINE</code> | Underlining | | | | | | | | | | | | | | |
| <code>A_REVERSE</code> | Reverse video | | | | | | | | | | | | | | |
| <code>A_BLINK</code> | Blinking | | | | | | | | | | | | | | |
| <code>A_DIM</code> | Half bright | | | | | | | | | | | | | | |
| <code>A_BOLD</code> | Extra bright or bold | | | | | | | | | | | | | | |
| <code>A_ALTCHARSET</code> | Alternate character set | | | | | | | | | | | | | | |

attrset(3CURSES)

A_CHARTEXT Bit-mask to extract a character

COLOR_PAIR(*n*) Color-pair number *n*

The following macro is the reverse of COLOR_PAIR(*n*) :

PAIR_NUMBER(*attrs*) Returns the pair number associated with the COLOR_PAIR(*n*) attribute

RETURN VALUES These routines always return 1.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_addch(3CURSES), curs_addstr(3CURSES), curs_printw(3CURSES), curses(3CURSES), attributes(5)

NOTES The header < curses.h > automatically includes the headers < stdio.h > and < unctrl.h >.

Note that attroff(), wattroff(), attron(), wattron(), wattrset(), standend(), and standout() may be macros.

attr_set(3XCURSES)

| | | | | | | | | | |
|--------------------|--|--------------|--|--------------|---------------------------------------|-------------|-----------------------------|------------|---|
| NAME | attr_get, attr_off, attr_on, attr_set, color_set, wattr_get, wattr_off, wattr_on, wattr_set, wcolor_set – control window attributes | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int attr_get(attr_t *attrs, short *color, void *opts); int attr_off(attr_t attrs, void *opts); int attr_on(attr_t attrs, void *opts); int attr_set(attr_t attrs, short color, void *opts); int color_set(short *color, void *opts); int wattr_get(WINDOW *win, attr_t attrs, short *color, void *opts); int wattr_off(WINDOW *win, attr_t attrs, void *opts); int wattr_on(WINDOW *win, attr_t attrs, void *opts); int wattr_set(WINDOW *win, attr_t attrs, short color, void *opts); int wcolor_set(WINDOW *win, short color, void *opts);</pre> | | | | | | | | |
| DESCRIPTION | <p>The attr_get() function retrieves the current rendition of <i>stdscr</i>. The wattr_get() function retrieves the current rendition of window <i>win</i>. If <i>attrs</i> or <i>color</i> is a null pointer, no information is retrieved.</p> <p>The attr_off() and attr_on() functions unset and set, respectively, the specified window attributes of <i>stdscr</i>. These functions only affect the attributes specified; attributes that existed before the call are retained.</p> <p>The wattr_off() and wattr_on() functions unset or set the specified attributes for window <i>win</i>.</p> <p>The attr_set() and wattr_set() functions change the rendition of <i>stdscr</i> and <i>win</i>; the old values are not retained.</p> <p>The color_set() and wcolor_set() functions set the window color of <i>stdscr</i> and <i>win</i> to <i>color</i>.</p> <p>The attributes and color pairs that can be used are specified in the Attributes, Color Pairs, and Renditions section of the curses(3XCURSES) man page.</p> | | | | | | | | |
| PARAMETERS | <table><tr><td><i>attrs</i></td><td>Is a pointer to the foreground window attributes to be set or unset.</td></tr><tr><td><i>color</i></td><td>Is a pointer to a color pair number .</td></tr><tr><td><i>opts</i></td><td>Is reserved for future use.</td></tr><tr><td><i>win</i></td><td>Is a pointer to the window in which attribute changes are to be made.</td></tr></table> | <i>attrs</i> | Is a pointer to the foreground window attributes to be set or unset. | <i>color</i> | Is a pointer to a color pair number . | <i>opts</i> | Is reserved for future use. | <i>win</i> | Is a pointer to the window in which attribute changes are to be made. |
| <i>attrs</i> | Is a pointer to the foreground window attributes to be set or unset. | | | | | | | | |
| <i>color</i> | Is a pointer to a color pair number . | | | | | | | | |
| <i>opts</i> | Is reserved for future use. | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which attribute changes are to be made. | | | | | | | | |

attr_set(3XCURSES)

RETURN VALUES These functions always return OK.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO add_wch(3XCURSES), addnwstr(3XCURSES), attroff(3XCURSES), bkgrndset(3XCURSES), curses(3XCURSES), init_color(3XCURSES), libcurses(3XCURSES), start_color(3XCURSES), attributes(5), standards(5)

attrset(3XCURSES)

| | |
|----------------------|--|
| NAME | attroff, attron, attrset, wattroff, wattron, wattrset – change foreground window attributes |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int attroff(int attrs); int attron(int attrs); int attrset(int attrs); int wattroff(WINDOW *win, int attrs); int wattron(WINDOW *win, int attrs); int wattrset(WINDOW *win, int attrs);</pre> |
| DESCRIPTION | <p>The <code>attroff()</code> and <code>attron()</code> functions unset and set, respectively, the specified window attributes of <code>stdscr</code>. These functions only affect the attributes specified; attributes that existed before the call are retained. The <code>wattroff()</code> and <code>wattron()</code> functions unset or set the specified attributes for window <code>win</code>.</p> <p>The <code>attrset()</code> and <code>wattrset()</code> functions change the specified window renditions of <code>stdscr</code> and <code>win</code> to new values; the old values are not retained.</p> <p>The attributes that can be used are specified in the <code>Attributes, Color Pairs, and Renditions</code> section of the <code>curses(3XCURSES)</code> man page.</p> <p>Here is an example that prints some text using the current window rendition, adds underlining, changes the attributes, prints more text, then changes the attributes back.</p> <pre>printw("This word is"); attron(A_UNDERLINE); printw("underlined."); attroff(A_NORMAL); printw("This is back to normal text.\n"); refresh();</pre> |
| PARAMETERS | <p><i>attrs</i> are the foreground window attributes to be set or unset.</p> <p><i>win</i> Is a pointer to the window in which attribute changes are to be made.</p> |
| RETURN VALUES | These functions always return OK or 1. |
| ERRORS | None. |
| USAGE | All of these functions may be macros. |

attrset(3XCURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addch(3XCURSES)`, `addnstr(3XCURSES)`, `attr_get(3XCURSES)`, `bkgdset(3XCURSES)`, `curses(3XCURSES)`, `init_color(3XCURSES)`, `libcurses(3XCURSES)`, `start_color(3XCURSES)`, `attributes(5)`, `standards(5)`

baudrate(3CURSES)

| | |
|----------------------|---|
| NAME | <code> curs_termattrs, baudrate, erasechar, has_ic, has_il, killchar, longname, termattrs, termname</code> – curses environment query routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int baudrate(void); char erasechar(void); int has_ic(void); int has_il(void); char killchar(void); char *longname(void); cctype termattrs(void); char *termname(void);</pre> |
| DESCRIPTION | <p>The <code>baudrate()</code> routine returns the output speed of the terminal. The number returned is in bits per second, for example 9600, and is an integer.</p> <p>With the <code>erasechar()</code> routine, the user's current erase character is returned.</p> <p>The <code>has_ic()</code> routine is true if the terminal has insert- and delete-character capabilities.</p> <p>The <code>has_il()</code> routine is true if the terminal has insert- and delete-line capabilities, or can simulate them using scrolling regions. This might be used to determine if it would be appropriate to turn on physical scrolling using <code>scrollok()</code>.</p> <p>With the <code>killchar()</code> routine, the user's current line kill character is returned.</p> <p>The <code>longname()</code> routine returns a pointer to a static area containing a verbose description of the current terminal. The maximum length of a verbose description is 128 characters. It is defined only after the call to <code>initscr()</code> or <code>newterm()</code>. The area is overwritten by each call to <code>newterm()</code> and is not restored by <code>set_term()</code>, so the value should be saved between calls to <code>newterm()</code> if <code>longname()</code> is going to be used with multiple terminals.</p> <p>If a given terminal doesn't support a video attribute that an application program is trying to use, <code>curses</code> may substitute a different video attribute for it. The <code>termattrs()</code> function returns a logical OR of all video attributes supported by the terminal. This information is useful when a <code>curses</code> program needs complete control over the appearance of the screen.</p> <p>The <code>termname()</code> routine returns the value of the environment variable <code>TERM</code> (truncated to 14 characters).</p> |
| RETURN VALUES | <code>longname()</code> and <code>termname()</code> return <code>NULL</code> on error. |

baudrate(3CURSES)

Routines that return an integer return ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_initscr(3CURSES), curs_outopts(3CURSES), curses(3CURSES), attributes(5)

NOTES The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.

Note that termattrs() may be a macro.

baudrate(3XCURSES)

| NAME | baudrate – return terminal baud rate | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int baudrate(void);</pre> | | | | | | |
| DESCRIPTION | The baudrate() function returns the terminal's data communication line and output speed in bits per second (for example, 9600). | | | | | | |
| RETURN VALUES | The baudrate() function returns the output speed of the terminal. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Standard</td></tr><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | libcurses(3XCURSES), attributes(5), standards(5) | | | | | | |

| NAME | curs_beep, beep, flash – curses bell and screen flash routines | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int beep(void); int flash(void);</pre> | | | | |
| DESCRIPTION | <p>The <code>beep()</code> and <code>flash()</code> routines are used to signal the terminal user. The routine <code>beep()</code> sounds the audible alarm on the terminal, if possible; if that is not possible, it flashes the screen (visible bell), if that is possible. The routine <code>flash()</code> flashes the screen, and if that is not possible, sounds the audible signal. If neither signal is possible, nothing happens. Nearly all terminals have an audible signal (bell or beep), but only some can flash the screen.</p> | | | | |
| RETURN VALUES | These routines always return OK. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . | | | | |

beep(3XCURSES)

NAME beep, flash – activate audio-visual alarm

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

int beep(void);

int flash(void);
```

DESCRIPTION The `beep()` and `flash()` functions produce an audio and visual alarm on the terminal, respectively. If the terminal has the capability, `beep()` sounds a bell or beep and `flash()` flashes the screen. One alarm is substituted for another if the terminal does not support the capability called (see `terminfo(4)` `bel` and `flash` capabilities). For example, a call to `beep()` for a terminal without that capability results in a flash.

RETURN VALUES These functions always return OK.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `libcurses(3XCURSES)`, `terminfo(4)`, `attributes(5)`, `standards(5)`

| NAME | curs_bkgd, bkgd, bkgdset, wbkgdset, wbkgd – curses window background manipulation routines | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int bkgd(chtype <i>ch</i>); void bkgdset(chtype <i>ch</i>); void wbkgdset(WINDOW *<i>win</i>, chtype <i>ch</i>); int wbkgd(WINDOW *<i>win</i>, chtype <i>ch</i>);</pre> | | | | |
| DESCRIPTION | <p>The <code>bkgdsets()</code> and <code>wbkgdset()</code> routines manipulate the background of the named window. Background is a <code>chtype</code> consisting of any combination of attributes and a character. The attribute part of the background is combined (ORed) with all non-blank characters that are written into the window with <code>waddch()</code>. Both the character and attribute parts of the background are combined with the blank characters. The background becomes a property of the character and moves with the character through any scrolling and insert/delete line/character operations. To the extent possible on a particular terminal, the attribute part of the background is displayed as the graphic rendition of the character put on the screen.</p> <p>The <code>bkgd()</code> and <code>wbkgd()</code> routines combine the new background with every position in the window. Background is any combination of attributes and a character. Only the attribute part is used to set the background of non-blank characters, while both character and attributes are used for blank positions. To the extent possible on a particular terminal, the attribute part of the background is displayed as the graphic rendition of the character put on the screen.</p> | | | | |
| RETURN VALUES | <code>bkgd()</code> and <code>wbkgd()</code> return the integer <code>OK</code> , or a non-negative integer, if <code>immedok()</code> is set. See <code>curs_ouptops(3CURSES)</code> . | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_addch(3CURSES)</code> , <code>curs_ouptops(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that <code>bkgdset()</code> and <code>bkgd()</code> may be macros.</p> | | | | |

bkgd(3XCURSES)

| | |
|----------------------|---|
| NAME | <code>bkgd</code> , <code>bkgdset</code> , <code>getbkgd</code> , <code>wbkgd</code> , <code>wbkgdset</code> – set or get the background character (and rendition) of window |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int bkgd(chtype <i>ch</i>); void bkgdset(chtype <i>ch</i>); chtype getbkgd(WINDOW *<i>win</i>); int wbkgd(WINDOW *<i>win</i>, chtype <i>ch</i>); void wbkgdset(WINDOW *<i>win</i>, chtype <i>ch</i>);</pre> |
| DESCRIPTION | <p>The <code>bkgdset()</code> and <code>wbkgdset()</code> functions turn off the previous background attributes, logical OR the requested attributes into the window rendition, and set the background property of the current or specified window based on the information in <i>ch</i>. If <i>ch</i> refers to a multi-column character, the results are undefined.</p> <p>The <code>bkgd()</code> and <code>wbkgd()</code> functions turn off the previous background attributes, logical OR the requested attributes into the window rendition, and set the background property of the current or specified window and then apply this setting to every character position in that window:</p> <ul style="list-style-type: none">■ The rendition of every character on the screen is changed to the new window rendition.■ Wherever the former background character appears, it is changed to the new background character. <p>The <code>getbkgd()</code> function extracts the specified window's background character and rendition.</p> |
| PARAMETERS | <p><i>ch</i> Is the background character to be set.</p> <p><i>win</i> Is a pointer to the window in which the background character is to be set.</p> |
| RETURN VALUES | <p>Upon successful completion, the <code>bkgd()</code> and <code>wbkgd()</code> functions return OK. Otherwise, they return ERR.</p> <p>The <code>bkgdset()</code> and <code>wbkgdset()</code> functions do not return a value.</p> <p>Upon successful completion, the <code>getbkgd()</code> function returns the specified window's background character and rendition. Otherwise, it returns (chtype) ERR.</p> |
| ERRORS | No errors are defined. |

bkgd(3XCURSES)

USAGE These functions are only guaranteed to operate reliably on character sets in which each character fits into a single byte, whose attributes can be expressed using only constants with the A_ prefix.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO addch(3XCURSES), addchstr(3XCURSES), attroff(3XCURSES), bkgrnd(3XCURSES), clear(3XCURSES), clrtoeol(3XCURSES), clrtoeol(3XCURSES), clrtobot(3XCURSES), erase(3XCURSES), inch(3XCURSES), libcurses(3XCURSES), mvprintw(3XCURSES), attributes(5), standards(5)

bkgdset(3CURSES)

| NAME | <code> curs_bkgd, bkgd, bkgdset, wbkgdset, wbkgd</code> – curses window background manipulation routines | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int bkgd(chtype <i>ch</i>); void bkgdset(chtype <i>ch</i>); void wbkgdset(WINDOW *<i>win</i>, chtype <i>ch</i>); int wbkgd(WINDOW *<i>win</i>, chtype <i>ch</i>);</pre> | | | | |
| DESCRIPTION | <p>The <code>bkgdset()</code> and <code>wbkgdset()</code> routines manipulate the background of the named window. Background is a <code>chtype</code> consisting of any combination of attributes and a character. The attribute part of the background is combined (ORed) with all non-blank characters that are written into the window with <code>waddch()</code>. Both the character and attribute parts of the background are combined with the blank characters. The background becomes a property of the character and moves with the character through any scrolling and insert/delete line/character operations. To the extent possible on a particular terminal, the attribute part of the background is displayed as the graphic rendition of the character put on the screen.</p> <p>The <code>bkgd()</code> and <code>wbkgd()</code> routines combine the new background with every position in the window. Background is any combination of attributes and a character. Only the attribute part is used to set the background of non-blank characters, while both character and attributes are used for blank positions. To the extent possible on a particular terminal, the attribute part of the background is displayed as the graphic rendition of the character put on the screen.</p> | | | | |
| RETURN VALUES | <code>bkgd()</code> and <code>wbkgd()</code> return the integer <code>OK</code> , or a non-negative integer, if <code>immedok()</code> is set. See <code>curls_ouptos(3CURSES)</code> . | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curls_addch(3CURSES)</code> , <code>curls_ouptos(3CURSES)</code> , <code>curls(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that <code>bkgdset()</code> and <code>bkgd()</code> may be macros.</p> | | | | |

| | |
|----------------------|--|
| NAME | bkgd, bkgdset, getbkgd, wbkgd, wbkgdset – set or get the background character (and rendition) of window |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int bkgd (chtype ch); void bkgdset (chtype ch); chtype getbkgd (WINDOW *win); int wbkgd (WINDOW *win, chtype ch); void wbkgdset (WINDOW *win, chtype ch);</pre> |
| DESCRIPTION | <p>The bkgdset () and wbkgdset () functions turn off the previous background attributes, logical OR the requested attributes into the window rendition, and set the background property of the current or specified window based on the information in <i>ch</i>. If <i>ch</i> refers to a multi-column character, the results are undefined.</p> <p>The bkgd () and wbkgd () functions turn off the previous background attributes, logical OR the requested attributes into the window rendition, and set the background property of the current or specified window and then apply this setting to every character position in that window:</p> <ul style="list-style-type: none"> ■ The rendition of every character on the screen is changed to the new window rendition. ■ Wherever the former background character appears, it is changed to the new background character. <p>The getbkgd () function extracts the specified window's background character and rendition.</p> |
| PARAMETERS | <p><i>ch</i> Is the background character to be set.</p> <p><i>win</i> Is a pointer to the window in which the background character is to be set.</p> |
| RETURN VALUES | <p>Upon successful completion, the bkgd () and wbkgd () functions return OK. Otherwise, they return ERR.</p> <p>The bkgdset () and wbkgdset () functions do not return a value.</p> <p>Upon successful completion, the getbkgd () function returns the specified window's background character and rendition. Otherwise, it returns (chtype) ERR.</p> |
| ERRORS | No errors are defined. |

bkgdset(3XCURSES)

USAGE These functions are only guaranteed to operate reliably on character sets in which each character fits into a single byte, whose attributes can be expressed using only constants with the A_ prefix.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addch(3XCURSES)`, `addchstr(3XCURSES)`, `attroff(3XCURSES)`, `bkgrnd(3XCURSES)`, `clear(3XCURSES)`, `clrtoeol(3XCURSES)`, `clrtobot(3XCURSES)`, `erase(3XCURSES)`, `inch(3XCURSES)`, `libcurses(3XCURSES)`, `mvprintw(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|----------------------|---|
| NAME | bkgrnd, bkgrndset, getbkgrnd, wbkgrnd, wbkgrndset, wgetbkgrnd – set or get the background character (and rendition) of window using a complex character |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int bkgrnd(const cchar_t *wch); void bkgrndset(const cchar_t *wch); int getbkgrnd(cchar_t *wch); int wbkgrnd(WINDOW *win, const cchar_t *wch); void wbkgrndset(WINDOW *win, const cchar_t *wch); int wgetbkgrnd(WINDOW *win, cchar_t *wch);</pre> |
| DESCRIPTION | <p>The bkgrndset () and wbkgrndset () functions turn off the previous background attributes, logical OR the requested attributes into the window rendition, and set the background property of the current or specified window based on the information in <i>wch</i>.</p> <p>The bkgrnd () and wbkgrnd () functions turn off the previous background attributes, logical OR the requested attributes into the window rendition, and set the background property of the current or specified window and then apply this setting to every character position in that window:</p> <ul style="list-style-type: none"> ■ The rendition of every character on the screen is changed to the new window rendition. ■ Wherever the former background character appears, it is changed to the new background character. <p>If <i>wch</i> refers to a non-spacing complex character for bkgrnd (), bkgrndset (), wbkgrnd (), and wbkgrndset (), then <i>wch</i> is added to the existing spacing complex character that is the background character. If <i>wch</i> refers to a multi-column character, the results are unspecified.</p> <p>The getbkgrnd () and wgetbkgrnd () functions store, into the area pointed to by <i>wch</i>, the window's background character and rendition.</p> |
| PARAMETERS | <p><i>wch</i> Is a pointer to the complex background character to be set.</p> <p><i>win</i> Is a pointer to the window in which the complex background character is to be set.</p> |
| RETURN VALUES | <p>The bkgrndset () and wbkgrndset () functions do not return a value.</p> <p>Upon successful completion, the other functions return OK. Otherwise, they return ERR.</p> |

bkgnd(3XCURSES)

ERRORS No errors are defined.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `add_wch(3XCURSES)`, `add_wchnstr(3XCURSES)`, `addch(3XCURSES)`, `addchstr(3XCURSES)`, `attroff(3XCURSES)`, `bkgd(3XCURSES)`, `clear(3XCURSES)`, `clrtoeol(3XCURSES)`, `clrtobot(3XCURSES)`, `erase(3XCURSES)`, `inch(3XCURSES)`, `libcurses(3XCURSES)`, `mvprintw(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|----------------------|---|
| NAME | bkgrnd, bkgrndset, getbkgrnd, wbkgrnd, wbkgrndset, wgetbkgrnd – set or get the background character (and rendition) of window using a complex character |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int bkgrnd(const cchar_t *wch); void bkgrndset(const cchar_t *wch); int getbkgrnd(cchar_t *wch); int wbkgrnd(WINDOW *win, const cchar_t *wch); void wbkgrndset(WINDOW *win, const cchar_t *wch); int wgetbkgrnd(WINDOW *win, cchar_t *wch);</pre> |
| DESCRIPTION | <p>The <code>bkgrndset()</code> and <code>wbkgrndset()</code> functions turn off the previous background attributes, logical OR the requested attributes into the window rendition, and set the background property of the current or specified window based on the information in <code>wch</code>.</p> <p>The <code>bkgrnd()</code> and <code>wbkgrnd()</code> functions turn off the previous background attributes, logical OR the requested attributes into the window rendition, and set the background property of the current or specified window and then apply this setting to every character position in that window:</p> <ul style="list-style-type: none"> ■ The rendition of every character on the screen is changed to the new window rendition. ■ Wherever the former background character appears, it is changed to the new background character. <p>If <code>wch</code> refers to a non-spacing complex character for <code>bkgrnd()</code>, <code>bkgrndset()</code>, <code>wbkgrnd()</code>, and <code>wbkgrndset()</code>, then <code>wch</code> is added to the existing spacing complex character that is the background character. If <code>wch</code> refers to a multi-column character, the results are unspecified.</p> <p>The <code>getbkgrnd()</code> and <code>wgetbkgrnd()</code> functions store, into the area pointed to by <code>wch</code>, the window's background character and rendition.</p> |
| PARAMETERS | <p><code>wch</code> Is a pointer to the complex background character to be set.</p> <p><code>win</code> Is a pointer to the window in which the complex background character is to be set.</p> |
| RETURN VALUES | <p>The <code>bkgrndset()</code> and <code>wbkgrndset()</code> functions do not return a value.</p> <p>Upon successful completion, the other functions return <code>OK</code>. Otherwise, they return <code>ERR</code>.</p> |

bkgrndset(3XCURSES)

ERRORS No errors are defined.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `add_wch(3XCURSES)`, `add_wchnstr(3XCURSES)`, `addch(3XCURSES)`, `addchstr(3XCURSES)`, `attroff(3XCURSES)`, `bkgd(3XCURSES)`, `clear(3XCURSES)`, `clrtoeol(3XCURSES)`, `clrtobot(3XCURSES)`, `erase(3XCURSES)`, `inch(3XCURSES)`, `libcurses(3XCURSES)`, `mvprintw(3XCURSES)`, `attributes(5)`, `standards(5)`

| | | | | | | | | | | | | | | | | | |
|--------------------|---|-----------|-------------------------|-----------|--------------------------|-----------|------------------------|-----------|---------------------------|-----------|----------------------|-----------|-----------------------|-----------|-------------------------|-----------|--------------------------|
| NAME | curs_border, border, wborder, box, whline, wvline – create curses borders, horizontal and vertical lines | | | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int border(chtype ls, chtype rs, chtype ts, chtype bs, chtype tl, chtype tr, chtype bl, chtype br); int wborder(WINDOW *win, chtype ls, chtype rs, chtype ts, chtype bs, chtype tl, chtype tr, chtype bl, chtype br); int box(WINDOW *win, chtype verch, chtype horch); int hline(chtype ch, int n); int whline(WINDOW *win, chtype ch, int n); int vline(chtype ch, int n); int wvline(WINDOW *win, chtype ch, int n);</pre> | | | | | | | | | | | | | | | | |
| DESCRIPTION | <p>With the <code>border()</code>, <code>wborder()</code>, and <code>box()</code> routines, a border is drawn around the edges of the window. The arguments and attributes are:</p> <hr/> <table border="0"> <tr> <td style="padding-right: 20px;"><i>ls</i></td> <td>left side of the border</td> </tr> <tr> <td><i>rs</i></td> <td>right side of the border</td> </tr> <tr> <td><i>ts</i></td> <td>top side of the border</td> </tr> <tr> <td><i>bs</i></td> <td>bottom side of the border</td> </tr> <tr> <td><i>tl</i></td> <td>top left-hand corner</td> </tr> <tr> <td><i>tr</i></td> <td>top right-hand corner</td> </tr> <tr> <td><i>bl</i></td> <td>bottom left-hand corner</td> </tr> <tr> <td><i>br</i></td> <td>bottom right-hand corner</td> </tr> </table> <hr/> <p>If any of these arguments is zero, then the following default values (defined in <code><curses.h></code>) are used respectively instead: <code>ACS_VLINE</code>, <code>ACS_VLINE</code>, <code>ACS_HLINE</code>, <code>ACS_HLINE</code>, <code>ACS_ULCORNER</code>, <code>ACS_URCORNER</code>, <code>ACS_BLCORNER</code>, <code>ACS_BRCORNER</code>.</p> <p><code>box(win, verch, horch)</code> is a shorthand for the following call:</p> <pre>wborder(win, verch, verch, horch, horch, 0, 0, 0, 0)</pre> <p><code>hline()</code> and <code>whline()</code> draw a horizontal (left to right) line using <i>ch</i> starting at the current cursor position in the window. The current cursor position is not changed. The line is at most <i>n</i> characters long, or as many as fit into the window.</p> | <i>ls</i> | left side of the border | <i>rs</i> | right side of the border | <i>ts</i> | top side of the border | <i>bs</i> | bottom side of the border | <i>tl</i> | top left-hand corner | <i>tr</i> | top right-hand corner | <i>bl</i> | bottom left-hand corner | <i>br</i> | bottom right-hand corner |
| <i>ls</i> | left side of the border | | | | | | | | | | | | | | | | |
| <i>rs</i> | right side of the border | | | | | | | | | | | | | | | | |
| <i>ts</i> | top side of the border | | | | | | | | | | | | | | | | |
| <i>bs</i> | bottom side of the border | | | | | | | | | | | | | | | | |
| <i>tl</i> | top left-hand corner | | | | | | | | | | | | | | | | |
| <i>tr</i> | top right-hand corner | | | | | | | | | | | | | | | | |
| <i>bl</i> | bottom left-hand corner | | | | | | | | | | | | | | | | |
| <i>br</i> | bottom right-hand corner | | | | | | | | | | | | | | | | |

border(3CURSES)

`vline()` and `wvline()` draw a vertical (top to bottom) line using `ch` starting at the current cursor position in the window. The current cursor position is not changed. The line is at most *n* characters long, or as many as fit into the window.

RETURN VALUES All routines return the integer `OK`, or a non-negative integer if `immedok()` is set. See `curl_ouptos(3CURSES)`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_ouptos(3CURSES)`, `curls(3CURSES)`, `attributes(5)`

NOTES The header `<curls.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `border()` and `box()` may be macros.

NAME border, box, wborder – add a single-byte border to a window

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

int border(chtype ls, chtype rs, chtype ts, chtype bs, chtype tl,
           chtype tr, chtype bl, chtype br);

int wborder(WINDOW *win, chtype ls, chtype rs, chtype ts, chtype bs,
            chtype tl, chtype tr, chtype bl, chtype br);

int box(WINDOW *win, chtype verch, chtype horch);
```

DESCRIPTION The `border()` and `wborder()` functions draw a border around the specified window. All parameters must be single-byte characters whose rendition can be expressed using only constants beginning with `ACS_`. A parameter with the value of 0 is replaced by the default value.

| Constant Values for Borders | | |
|-----------------------------|------------------|-------------------|
| Parameter | Default Constant | Default Character |
| <i>verch</i> | ACS_VLINE | |
| <i>horch</i> | ACS_HLINE | - |
| <i>ls</i> | ACS_VLINE | |
| <i>rs</i> | ACS_VLINE | |
| <i>ts</i> | ACS_HLINE | - |
| <i>bs</i> | ACS_HLINE | - |
| <i>bl</i> | ACS_BLCORNER | + |
| <i>br</i> | ACS_BRCORNER | + |
| <i>tl</i> | ACS_ULCORNER | + |
| <i>tr</i> | ACS_URCORNER | + |

The call

`box(win,
verch, horch)` is a short form for

```
wborder(win,
        verch, verch,
        horch, horch, 0, 0, 0,
```

border(3XCURSES)

o) When the window is boxed, the bottom and top rows and right and left columns overwrite existing text.

PARAMETERS

ls Is the character and rendition used for the left side of the border.
rs Is the character and rendition used for the right side of the border.
ts Is the character and rendition used for the top of the border.
bs Is the character and rendition used for the bottom of the border.
tl Is the character and rendition used for the top-left corner of the border.
tr Is the character and rendition used for the top-right corner of the border.
bl Is the character and rendition used for the bottom-left corner of the border.
br Is the character and rendition used for the bottom-right corner of the border.
win Is the pointer to the window in which the border or box is to be drawn.
verch Is the character and rendition used for the left and right columns of the box.
horch Is the character and rendition used for the top and bottom rows of the box.

RETURN VALUES

On success, these functions return OK. Otherwise, they return ERR.

ERRORS

None.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO

`add_wch(3XCURSES)`, `addch(3XCURSES)`, `attr_get(3XCURSES)`,
`attroff(3XCURSES)`, `border_set(3XCURSES)`, `libcurses(3XCURSES)`,
`attributes(5)`, `standards(5)`

NAME border_set, box_set, wborder_set – use complex characters (and renditions) to draw borders

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

int border_set(const cchar_t *ls, const cchar_t *rs, const cchar_t
               *ts, const cchar_t *bs, const cchar_t *tl, const cchar_t *tr,
               const cchar_t *bl, const cchar_t *br);

int wborder_set(WINDOW *win, const cchar_t *ls, const cchar_t *rs,
               const cchar_t *ts, const cchar_t *bs, const cchar_t *tl, const
               cchar_t *tr, const cchar_t *bl, const cchar_t *br);

int box_set(WINDOW *win, const cchar_t *verch, const cchar_t
            *horch);
```

DESCRIPTION The border_set () and wborder_set () functions draw a border around the specified window. All parameters must be spacing complex characters with renditions. A parameter which is a null pointer is replaced by the default character.

Constant Values for Borders

| Constant Values for Borders | | |
|-----------------------------|------------------|-------------------|
| Parameter | Default Constant | Default Character |
| <i>verch</i> | WACS_VLINE | |
| <i>horch</i> | WACS_HLINE | - |
| <i>ls</i> | WACS_VLINE | |
| <i>rs</i> | WACS_VLINE | |
| <i>ts</i> | WACS_HLINE | - |
| <i>bs</i> | WACS_HLINE | - |
| <i>bl</i> | WACS_BLCORNER | + |
| <i>br</i> | WACS_BRCORNER | + |
| <i>tl</i> | WACS_ULCORNER | + |
| <i>tr</i> | WACS_URCORNER | + |

The call

border_set(3XCURSES)

`box_set(win, verch, horch)` is a short form for

```
wborder(win, verch, verch, horch, horch, NULL, NULL, NULL, NULL)
```

When the window is boxed, the bottom and top rows and right and left columns are unavailable for text.

PARAMETERS

ls Is the character and rendition used for the left side of the border.

rs Is the character and rendition used for the right side of the border.

ts Is the character and rendition used for the top of the border.

bs Is the character and rendition used for the bottom of the border.

tl Is the character and rendition used for the top-left corner of the border.

tr Is the character and rendition used for the top-right corner of the border.

bl Is the character and rendition used for the bottom-left corner of the border.

br Is the character and rendition used for the bottom-right corner of the border.

win Is the pointer to the window in which the border or box is to be drawn.

verch Is the character and rendition used for the left and right columns of the box.

horch Is the character and rendition used for the top and bottom rows of the box.

RETURN VALUES

On success, these functions return OK. Otherwise, they return ERR.

ERRORS

None.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

`border_set(3XCURSES)`

SEE ALSO `add_wch(3XCURSES)`, `addch(3XCURSES)`, `attr_get(3XCURSES)`,
`attroff(3XCURSES)`, `border(3XCURSES)`, `libcurses(3XCURSES)`,
`attributes(5)`, `standards(5)`

bottom_panel(3CURSES)

| NAME | panel_top, top_panel, bottom_panel – panels deck manipulation routines | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lpanel -lcurses [library ..] #include <panel.h> int top_panel (PANEL *panel) ; int bottom_panel (PANEL *panel) ;</pre> | | | | |
| DESCRIPTION | <p>top_panel () pulls <i>panel</i> to the top of the desk of panels. It leaves the size, location, and contents of its associated window unchanged.</p> <p>bottom_panel () puts <i>panel</i> at the bottom of the deck of panels. It leaves the size, location, and contents of its associated window unchanged.</p> | | | | |
| RETURN VALUES | All of these routines return the integer OK upon successful completion or ERR upon error. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), panel_update(3CURSES), panels(3CURSES), attributes(5) | | | | |
| NOTES | The header <panel.h> automatically includes the header <curses.h>. | | | | |

| | | | | | | | | | | | | | | | | | |
|--------------------|---|-----------|-------------------------|-----------|--------------------------|-----------|------------------------|-----------|---------------------------|-----------|----------------------|-----------|-----------------------|-----------|-------------------------|-----------|--------------------------|
| NAME | curs_border, border, wborder, box, whline, wvline – create curses borders, horizontal and vertical lines | | | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int border(chtype <i>ls</i>, chtype <i>rs</i>, chtype <i>ts</i>, chtype <i>bs</i>, chtype <i>tl</i>, chtype <i>tr</i>, chtype <i>bl</i>, chtype <i>br</i>); int wborder(WINDOW *<i>win</i>, chtype <i>ls</i>, chtype <i>rs</i>, chtype <i>ts</i>, chtype <i>bs</i>, chtype <i>tl</i>, chtype <i>tr</i>, chtype <i>bl</i>, chtype <i>br</i>); int box(WINDOW *<i>win</i>, chtype <i>verch</i>, chtype <i>horch</i>); int hline(chtype <i>ch</i>, int <i>n</i>); int whline(WINDOW *<i>win</i>, chtype <i>ch</i>, int <i>n</i>); int vline(chtype <i>ch</i>, int <i>n</i>); int wvline(WINDOW *<i>win</i>, chtype <i>ch</i>, int <i>n</i>);</pre> | | | | | | | | | | | | | | | | |
| DESCRIPTION | <p>With the <code>border()</code>, <code>wborder()</code>, and <code>box()</code> routines, a border is drawn around the edges of the window. The arguments and attributes are:</p> <hr/> <table border="0"> <tr> <td style="padding-right: 20px;"><i>ls</i></td> <td>left side of the border</td> </tr> <tr> <td><i>rs</i></td> <td>right side of the border</td> </tr> <tr> <td><i>ts</i></td> <td>top side of the border</td> </tr> <tr> <td><i>bs</i></td> <td>bottom side of the border</td> </tr> <tr> <td><i>tl</i></td> <td>top left-hand corner</td> </tr> <tr> <td><i>tr</i></td> <td>top right-hand corner</td> </tr> <tr> <td><i>bl</i></td> <td>bottom left-hand corner</td> </tr> <tr> <td><i>br</i></td> <td>bottom right-hand corner</td> </tr> </table> <hr/> <p>If any of these arguments is zero, then the following default values (defined in <code><curses.h></code>) are used respectively instead: <code>ACS_VLINE</code>, <code>ACS_VLINE</code>, <code>ACS_HLINE</code>, <code>ACS_HLINE</code>, <code>ACS_ULCORNER</code>, <code>ACS_URCORNER</code>, <code>ACS_BLCORNER</code>, <code>ACS_BRCORNER</code>.</p> <p><code>box(<i>win</i>, <i>verch</i>, <i>horch</i>)</code> is a shorthand for the following call:</p> <pre>wborder(<i>win</i>, <i>verch</i>, <i>verch</i>, <i>horch</i>, <i>horch</i>, 0, 0, 0, 0)</pre> <p><code>hline()</code> and <code>whline()</code> draw a horizontal (left to right) line using <i>ch</i> starting at the current cursor position in the window. The current cursor position is not changed. The line is at most <i>n</i> characters long, or as many as fit into the window.</p> | <i>ls</i> | left side of the border | <i>rs</i> | right side of the border | <i>ts</i> | top side of the border | <i>bs</i> | bottom side of the border | <i>tl</i> | top left-hand corner | <i>tr</i> | top right-hand corner | <i>bl</i> | bottom left-hand corner | <i>br</i> | bottom right-hand corner |
| <i>ls</i> | left side of the border | | | | | | | | | | | | | | | | |
| <i>rs</i> | right side of the border | | | | | | | | | | | | | | | | |
| <i>ts</i> | top side of the border | | | | | | | | | | | | | | | | |
| <i>bs</i> | bottom side of the border | | | | | | | | | | | | | | | | |
| <i>tl</i> | top left-hand corner | | | | | | | | | | | | | | | | |
| <i>tr</i> | top right-hand corner | | | | | | | | | | | | | | | | |
| <i>bl</i> | bottom left-hand corner | | | | | | | | | | | | | | | | |
| <i>br</i> | bottom right-hand corner | | | | | | | | | | | | | | | | |

box(3CURSES)

`vline()` and `wvline()` draw a vertical (top to bottom) line using `ch` starting at the current cursor position in the window. The current cursor position is not changed. The line is at most *n* characters long, or as many as fit into the window.

RETURN VALUES All routines return the integer `OK`, or a non-negative integer if `immedok()` is set. See `curs_ouptops(3CURSES)`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_ouptops(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `border()` and `box()` may be macros.

| | |
|--------------------|--|
| NAME | plot, arc, box, circle, closepl, closevt, cont, erase, label, line, linemod, move, openpl, openvt, point, space – graphics interface |
| SYNOPSIS | <pre>cc [flag ...] file ... -lplot [library...] #include <plot.h> void arc(short x0, short y0, short x1, short y1, short x2, short y2); void box(short x0, short y0, short x1, short y1); void circle(short x, short y, short r); void closepl(); void closevt(); void cont(short x, short y); void erase(); void label(char *s); void line(short x0, short y0, short x1, short y1); void linemod(char *s); void move(short x, short y); void openpl(); void openvt(); void point(short x, short y); void space(short x0, short y0, short x1, short y1);</pre> |
| DESCRIPTION | <p>These functions generate graphics output for a set of output devices. The format of the output is dependent upon which link editor option is used when the program is compiled and linked (see Link Editor).</p> <p>The term "current point" refers to the current setting for the x and y coordinates.</p> <p>The <code>arc()</code> function specifies a circular arc. The coordinates $(x0, y0)$ specify the center of the arc. The coordinates $(x1, y1)$ specify the starting point of the arc. The coordinates $(x2, y2)$ specify the end point of the circular arc.</p> <p>The <code>box()</code> function specifies a rectangle with coordinates $(x0, y0)$, $(x0, y1)$, $(x1, y0)$, and $(x1, y1)$. The current point is set to $(x1, y1)$.</p> <p>The <code>circle()</code> function specifies a circle with a center at the coordinates (x, y) and a radius of r.</p> <p>The <code>closevt()</code> and <code>closepl()</code> functions flush the output.</p> <p>The <code>cont()</code> function specifies a line beginning at the current point and ending at the coordinates (x, y). The current point is set to (x, y).</p> |

box(3PLOT)

The `erase()` function starts another frame of output.

The `label()` function places the null terminated string `s` so that the first character falls on the current point. The string is then terminated by a NEWLINE character.

The `line()` function draws a line starting at the coordinates (x_0, y_0) and ending at the coordinates (x_1, y_1) . The current point is set to (x_1, y_1) .

The `linemod()` function specifies the style for drawing future lines. `s` may contain one of the following: `dotted`, `solid`, `longdashed`, `shortdashed`, or `dotdashed`.

The `move()` function sets the current point to the coordinates (x, y) .

The `openpl()` or `openvt()` function must be called to open the device before any other plot functions are called.

The `point()` function plots the point given by the coordinates (x, y) . The current point is set to (x, y) .

The `space()` function specifies the size of the plotting area. The plot will be reduced or enlarged as necessary to fit the area specified. The coordinates (x_0, y_0) specify the lower left hand corner of the plotting area. The coordinates (x_1, y_1) specify the upper right hand corner of the plotting area.

Link Editor Various flavors of these functions exist for different output devices. They are obtained by using the following `ld(1)` options:

| | |
|---------------------|--|
| <code>-lplot</code> | device-independent graphics stream on standard output in the format described in <code>plot(4B)</code> |
| <code>-l300</code> | GSI 300 terminal |
| <code>-l300s</code> | GSI 300S terminal |
| <code>-l4014</code> | Tektronix 4014 terminal |
| <code>-l450</code> | GSI 450 terminal |
| <code>-lvt0</code> | |

FILES

| | |
|--|----------------------|
| <code>/usr/lib/libplot.a</code> | archive library |
| <code>/usr/lib/libplot.so.1</code> | shared object |
| <code>/usr/lib/sparcv9/libplot.so.1</code> | 64-bit shared object |
| <code>/usr/lib/lib300.a</code> | archive library |
| <code>/usr/lib/lib300.so.1</code> | shared object |

```

/usr/lib/sparcv9/lib300.so.1
  64-bit shared object
/usr/lib/lib300s.a
  archive library
/usr/lib/lib300s.so.1
  shared object
/usr/lib/sparcv9/lib300s.so.1
  64-bit shared object
/usr/lib/lib4014.a
  archive library
/usr/lib/lib4014.so.1
  shared object
/usr/lib/sparcv9/lib4014.so.1
  64-bit shared object
/usr/lib/lib450.a
  archive library
/usr/lib/lib450.so.1
  shared object
/usr/lib/sparcv9/lib450.so.1
  64-bit shared object
/usr/lib/libvt0.a
  archive library
/usr/lib/libvt0.so.1
  shared object
/usr/lib/sparcv9/libvt0.so.1
  64-bit shared object

```

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `graph(1)`, `ld(1)`, `libplot(3LIB)`, `plot(4B)`, `attributes(5)`

box(3XCURSES)

NAME border, box, wborder – add a single-byte border to a window

SYNOPSIS `cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \`
`-R /usr/xpg4/lib -lcurses [library...]`

`c89 [flag...] file... -lcurses [library...]`

```
#include <curses.h>

int border(chtype ls, chtype rs, chtype ts, chtype bs, chtype tl,
           chtype tr, chtype bl, chtype br);

int wborder(WINDOW *win, chtype ls, chtype rs, chtype ts, chtype bs,
            chtype tl, chtype tr, chtype bl, chtype br);

int box(WINDOW *win, chtype verch, chtype horch);
```

DESCRIPTION The `border()` and `wborder()` functions draw a border around the specified window. All parameters must be single-byte characters whose rendition can be expressed using only constants beginning with `ACS_`. A parameter with the value of 0 is replaced by the default value.

| Constant Values for Borders | | |
|-----------------------------|------------------|-------------------|
| Parameter | Default Constant | Default Character |
| <i>verch</i> | ACS_VLINE | |
| <i>horch</i> | ACS_HLINE | - |
| <i>ls</i> | ACS_VLINE | |
| <i>rs</i> | ACS_VLINE | |
| <i>ts</i> | ACS_HLINE | - |
| <i>bs</i> | ACS_HLINE | - |
| <i>bl</i> | ACS_BLCORNER | + |
| <i>br</i> | ACS_BRCORNER | + |
| <i>tl</i> | ACS_ULCORNER | + |
| <i>tr</i> | ACS_URCORNER | + |

The call

`box(win,
verch, horch)` is a short form for

```
wborder(win,  
verch, verch,  
horch, horch, 0, 0, 0,
```

box(3XCURSES)

o) When the window is boxed, the bottom and top rows and right and left columns overwrite existing text.

PARAMETERS

ls Is the character and rendition used for the left side of the border.
rs Is the character and rendition used for the right side of the border.
ts Is the character and rendition used for the top of the border.
bs Is the character and rendition used for the bottom of the border.
tl Is the character and rendition used for the top-left corner of the border.
tr Is the character and rendition used for the top-right corner of the border.
bl Is the character and rendition used for the bottom-left corner of the border.
br Is the character and rendition used for the bottom-right corner of the border.
win Is the pointer to the window in which the border or box is to be drawn.
verch Is the character and rendition used for the left and right columns of the box.
horch Is the character and rendition used for the top and bottom rows of the box.

RETURN VALUES

On success, these functions return OK. Otherwise, they return ERR.

ERRORS

None.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO

`add_wch(3XCURSES)`, `addch(3XCURSES)`, `attr_get(3XCURSES)`, `attroff(3XCURSES)`, `border_set(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

box_set(3XCURSES)

NAME border_set, box_set, wborder_set – use complex characters (and renditions) to draw borders

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

int border_set(const cchar_t *ls, const cchar_t *rs, const cchar_t
    *ts, const cchar_t *bs, const cchar_t *tl, const cchar_t *tr,
    const cchar_t *bl, const cchar_t *br);

int wborder_set(WINDOW *win, const cchar_t *ls, const cchar_t *rs,
    const cchar_t *ts, const cchar_t *bs, const cchar_t *tl, const
    cchar_t *tr, const cchar_t *bl, const cchar_t *br);

int box_set(WINDOW *win, const cchar_t *verch, const cchar_t
    *horch);
```

DESCRIPTION The border_set() and wborder_set() functions draw a border around the specified window. All parameters must be spacing complex characters with renditions. A parameter which is a null pointer is replaced by the default character.

Constant Values for Borders

| Constant Values for Borders | | |
|-----------------------------|------------------|-------------------|
| Parameter | Default Constant | Default Character |
| <i>verch</i> | WACS_VLINE | |
| <i>horch</i> | WACS_HLINE | - |
| <i>ls</i> | WACS_VLINE | |
| <i>rs</i> | WACS_VLINE | |
| <i>ts</i> | WACS_HLINE | - |
| <i>bs</i> | WACS_HLINE | - |
| <i>bl</i> | WACS_BLCORNER | + |
| <i>br</i> | WACS_BRCORNER | + |
| <i>tl</i> | WACS_ULCORNER | + |
| <i>tr</i> | WACS_URCORNER | + |

The call

`box_set(win, verch, horch)` is a short form for

```
wborder(win,
verch, verch,
horch, horch, NULL,
NULL, NULL, NULL)
```

When the window is boxed, the bottom and top rows and right and left columns are unavailable for text.

PARAMETERS

- ls* Is the character and rendition used for the left side of the border.
- rs* Is the character and rendition used for the right side of the border.
- ts* Is the character and rendition used for the top of the border.
- bs* Is the character and rendition used for the bottom of the border.
- tl* Is the character and rendition used for the top-left corner of the border.
- tr* Is the character and rendition used for the top-right corner of the border.
- bl* Is the character and rendition used for the bottom-left corner of the border.
- br* Is the character and rendition used for the bottom-right corner of the border.
- win* Is the pointer to the window in which the border or box is to be drawn.
- verch* Is the character and rendition used for the left and right columns of the box.
- horch* Is the character and rendition used for the top and bottom rows of the box.

RETURN VALUES

On success, these functions return OK. Otherwise, they return ERR.

ERRORS

None.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

box_set(3XCURSES)

SEE ALSO | add_wch(3XCURSES), addch(3XCURSES), attr_get(3XCURSES),
attroff(3XCURSES), border(3XCURSES), libcurses(3XCURSES),
attributes(5), standards(5)

| | |
|-----------------------------|--|
| NAME | curs_color, start_color, init_pair, init_color, has_colors, can_change_color, color_content, pair_content – curses color manipulation routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int start_color(void); int init_pair(short pair, short fg, short bg); int init_color(short color, short red, short green, short blue); bool has_colors(void); bool can_change_color(void); int color_content(short color, short *redp, short *greenp, short *bluep); int pair_content(short pair, short *fgp, short *bgp);</pre> |
| Overview | <p>curses provides routines that manipulate color on color alphanumeric terminals. To use these routines <code>start_color()</code> must be called, usually right after <code>initscr()</code>. See <code>curs_initscr(3CURSES)</code>. Colors are always used in pairs (referred to as color-pairs). A color-pair consists of a foreground color (for characters) and a background color (for the field on which the characters are displayed). A programmer initializes a color-pair with the routine <code>init_pair</code>. After it has been initialized, <code>COLOR_PAIR(n)</code>, a macro defined in <code><curses.h></code>, can be used in the same ways other video attributes can be used. If a terminal is capable of redefining colors, the programmer can use the routine <code>init_color()</code> to change the definition of a color. The routines <code>has_colors()</code> and <code>can_change_color()</code> return <code>TRUE</code> or <code>FALSE</code>, depending on whether the terminal has color capabilities and whether the programmer can change the colors. The routine <code>color_content()</code> allows a programmer to identify the amounts of red, green, and blue components in an initialized color. The routine <code>pair_content()</code> allows a programmer to find out how a given color-pair is currently defined.</p> |
| Routine Descriptions | <p>The <code>start_color()</code> routine requires no arguments. It must be called if the programmer wants to use colors, and before any other color manipulation routine is called. It is good practice to call this routine right after <code>initscr()</code>. <code>start_color()</code> initializes eight basic colors (black, red, green, yellow, blue, magenta, cyan, and white), and two global variables, <code>COLORS</code> and <code>COLOR_PAIRS</code> (respectively defining the maximum number of colors and color-pairs the terminal can support). It also restores the colors on the terminal to the values they had when the terminal was just turned on.</p> <p>The <code>init_pair()</code> routine changes the definition of a color-pair. It takes three arguments: the number of the color-pair to be changed, the foreground color number, and the background color number. The value of the first argument must be between 1 and <code>COLOR_PAIRS-1</code>. The value of the second and third arguments must be between 0 and <code>COLORS</code>. If the color-pair was previously initialized, the screen is refreshed and all occurrences of that color-pair is changed to the new definition.</p> |

can_change_color(3CURSES)

The `init_color()` routine changes the definition of a color. It takes four arguments: the number of the color to be changed followed by three RGB values (for the amounts of red, green, and blue components). The value of the first argument must be between 0 and `COLORS`. (See the section `COLORS` for the default color index.) Each of the last three arguments must be a value between 0 and 1000. When `init_color()` is used, all occurrences of that color on the screen immediately change to the new definition.

The `has_colors()` routine requires no arguments. It returns `TRUE` if the terminal can manipulate colors; otherwise, it returns `FALSE`. This routine facilitates writing terminal-independent programs. For example, a programmer can use it to decide whether to use color or some other video attribute.

The `can_change_color()` routine requires no arguments. It returns `TRUE` if the terminal supports colors and can change their definitions; other, it returns `FALSE`. This routine facilitates writing terminal-independent programs.

The `color_content()` routine gives users a way to find the intensity of the red, green, and blue (RGB) components in a color. It requires four arguments: the color number, and three addresses of `shorts` for storing the information about the amounts of red, green, and blue components in the given color. The value of the first argument must be between 0 and `COLORS`. The values that are stored at the addresses pointed to by the last three arguments are between 0 (no component) and 1000 (maximum amount of component).

The `pair_content()` routine allows users to find out what colors a given color-pair consists of. It requires three arguments: the color-pair number, and two addresses of `shorts` for storing the foreground and the background color numbers. The value of the first argument must be between 1 and `COLOR_PAIRS-1`. The values that are stored at the addresses pointed to by the second and third arguments are between 0 and `COLORS`.

Colors In `< curses.h >` the following macros are defined. These are the default colors. `curses` also assumes that `COLOR_BLACK` is the default background color for all terminals.

```
COLOR_BLACK
COLOR_RED
COLOR_GREEN
COLOR_YELLOW
COLOR_BLUE
COLOR_MAGENTA
COLOR_CYAN
COLOR_WHITE
```

RETURN VALUES All routines that return an integer return `ERR` upon failure and `OK` upon successful completion.

can_change_color(3CURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_attr(3CURSES)`, `curl_initscr(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `< curses.h >` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

can_change_color(3XCURSES)

| | |
|------------------------------|---|
| NAME | can_change_color, color_content, COLOR_PAIR, has_colors, init_color, init_pair, pair_content, PAIR_NUMBER, start_color, COLOR_PAIRS, COLORS – manipulate color information |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> bool can_change_color(void); int color_content(short color, short *red, short *green, short *blue); int COLOR_PAIR(int n); bool has_colors(void); int init_color(short color, short red, short green, short blue); int init_pair(short pair, short f, short b); int pair_content(short pair, short *f, short *b); int PAIR_NUMBER(int value); int start_color(void); extern int COLOR_PAIRS; extern int COLORS;</pre> |
| DESCRIPTION | These functions manipulate color on terminals that support color. |
| Querying Capabilities | The has_colors() function indicates whether the terminal is a color terminal. The can_change_color() function indicates whether the terminal is a color terminal on which colors can be redefined. |
| Initialization | <p>The start_color() function must be called to enable use of colors and before any color manipulation function is called. The function initializes eight basic colors (black, red, green, yellow, blue, magenta, cyan, and white) that can be specified by the color macros (such as COLOR_BLACK) defined in <curses.h>. The initial appearance of these colors is unspecified.</p> <p>The function also initializes two global external variables:</p> <ul style="list-style-type: none">■ COLORS defines the number of colors that the terminal supports. See Color Identification below. If COLORS is 0, the terminal does not support redefinition of colors and can_change_color() will return FALSE.■ COLOR_PAIRS defines the maximum number of color-pairs that the terminal supports. See User-defined Color Pairs below. |

can_change_color(3XCURSES)

Color Identification

The `start_color()` function also restores the colors on the terminal to terminal-specific initial values. The initial background color is assumed to be black for all terminals.

The `init_color()` function redefines color number *color*, on terminals that support the redefinition of colors, to have the red, green, and blue intensity components specified by *red*, *green*, and *blue*, respectively. Calling `init_color()` also changes all occurrences of the specified color on the screen to the new definition.

The `color_content()` function identifies the intensity components of color number *color*. It stores the red, green, and blue intensity components of this color in the addresses pointed to by *red*, *green*, and *blue*, respectively.

For both functions, the *color* argument must be in the range from 0 to and including `COLORS-1`. Valid intensity value range from 0 (no intensity component) up to and including 1000 (maximum intensity in that component).

User-defined Color Pairs

Calling `init_pair()` defines or redefines color-pair number *pair* to have foreground color *f* and background color *b*. Calling `init_pair()` changes any characters that were displayed in the color pair's old definition to the new definition and refreshes the screen.

After defining the color pair, the macro `COLOR_PAIR(n)` returns the value of color pair *n*. This value is the color attribute as it would be extracted from a `chtype`. Controversy, the macro `COLOR_NUMBER(value)` returns the color pair number associated with the color attribute *value*.

The `pair_content()` retrieves the component colors of a color-pair number *pair*. It stores the foreground and background color numbers in the variables pointed to by *f* and *b*, respectively.

With `init_pair()` and `pair_content()`, the value of *pair* must be in a range from 0 to and including `COLOR_PAIRS-1`. Valid values for *f* and *b* are the range from 0 to and including `COLORS-1`.

PARAMETERS

| | |
|--------------|--|
| <i>color</i> | Is the number of the color for which to provide information (0 to <code>COLORS-1</code>). |
| <i>red</i> | Is a pointer to the RGB value for the amount of red in <i>color</i> . |
| <i>green</i> | Is a pointer to the RGB value for the amount of green in <i>color</i> . |
| <i>blue</i> | Is a pointer to the RGB value for the amount of blue in <i>color</i> . |
| <i>n</i> | Is the number of a color pair. |
| <i>pair</i> | Is the number of the color pair for which to provide information (1 to <code>COLOR_PAIRS-1</code>). |
| <i>f</i> | Is a pointer to the number of the foreground color (0 to <code>COLORS-1</code>) in <i>pair</i> . |

can_change_color(3XCURSES)

b Is a pointer to the number of the background color (0 to COLORS-1) in *pair*.

value Is a color attribute value.

RETURN VALUES The `has_colors()` function returns `TRUE` if the terminal can manipulate colors. Otherwise, it returns `FALSE`.

The `can_change_color()` function returns `TRUE` if the terminal supports colors and is able to change their definitions. Otherwise, it returns `FALSE`.

Upon successful completion, the other functions return `OK`. Otherwise, they return `ERR`.

ERRORS No errors are defined.

USAGE To use these functions, `start_color()` must be called, usually right after `initscr(3XCURSES)`.

The `can_change_color()` and `has_colors()` functions facilitate writing terminal-independent applications. For example, a programmer can use them to decide whether to use color or some other video attribute.

On color terminals, a typical value of `COLORS` is 8 and the macros such as `COLOR_BLACK` return a value within the range from 0 to and including 7. However, applications cannot rely on this to be true.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attroff(3XCURSES)`, `delscreen(3XCURSES)`, `initscr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | curs_inopts, cbreak, nocbreak, echo, noecho, halfdelay, intrflush, keypad, meta, nodelay, notimeout, raw, noraw, noqiflush, qiflush, timeout, wtimeout, typeahead – curses terminal input option control routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int cbreak(void); int nocbreak(void); int echo(void); int noecho(void); int halfdelay(int <i>tenths</i>); int intrflush(WINDOW *<i>win</i>, bool <i>bf</i>); int keypad(WINDOW *<i>win</i>, bool <i>bf</i>); int meta(WINDOW *<i>win</i>, bool <i>bf</i>); int nodelay(WINDOW *<i>win</i>, bool <i>bf</i>); int notimeout(WINDOW *<i>win</i>, bool <i>bf</i>); int raw(void); int noraw(void); void noqiflush(void); void qiflush(void); void timeout(int <i>delay</i>); void wtimeout(WINDOW *<i>win</i>, int <i>delay</i>); int typeahead(int <i>fildev</i>);</pre> |
| DESCRIPTION | <p>The <code>cbreak()</code> and <code>nocbreak()</code> routines put the terminal into and out of <code>cbreak()</code> mode, respectively. In this mode, characters typed by the user are immediately available to the program, and erase/kill character-processing is not performed. When out of this mode, the tty driver buffers the typed characters until a newline or carriage return is typed. Interrupt and flow control characters are unaffected by this mode. Initially the terminal may or may not be in <code>cbreak()</code> mode, as the mode is inherited; therefore, a program should call <code>cbreak()</code> or <code>nocbreak()</code> explicitly. Most interactive programs using <code>curses</code> set the <code>cbreak()</code> mode.</p> <p>Note that <code>cbreak()</code> overrides <code>raw()</code>. (See <code>curs_getch(3CURSES)</code> for a discussion of how these routines interact with <code>echo()</code> and <code>noecho()</code>.)</p> <p>The <code>echo()</code> and <code>noecho()</code> routines control whether characters typed by the user are echoed by <code>getch()</code> as they are typed. Echoing by the tty driver is always disabled, but initially <code>getch()</code> is in echo mode, so characters typed are echoed. Authors of</p> |

cbreak(3CURSES)

most interactive programs prefer to do their own echoing in a controlled area of the screen, or not to echo at all, so they disable echoing by calling `noecho()`. (See `curs_getch(3CURSES)` for a discussion of how these routines interact with `cbreak()` and `nocbreak()`.)

The `halfdelay()` routine is used for half-delay mode, which is similar to `cbreak()` mode in that characters typed by the user are immediately available to the program. However, after blocking for *tenths* tenths of seconds, `ERR` is returned if nothing has been typed. The value of *tenths* must be a number between 1 and 255. Use `nocbreak()` to leave half-delay mode.

If the `intrflush()` option is enabled, (*bf* is `TRUE`), when an interrupt key is pressed on the keyboard (interrupt, break, quit) all output in the tty driver queue will be flushed, giving the effect of faster response to the interrupt, but causing `curses` to have the wrong idea of what is on the screen. Disabling (*bf* is `FALSE`), the option prevents the flush. The default for the option is inherited from the tty driver settings. The window argument is ignored.

The `keypad()` option enables the keypad of the user's terminal. If enabled (*bf* is `TRUE`), the user can press a function key (such as an arrow key) and `wgetch()` returns a single value representing the function key, as in `KEY_LEFT`. If disabled (*bf* is `FALSE`), `curses` does not treat function keys specially and the program has to interpret the escape sequences itself. If the keypad in the terminal can be turned on (made to transmit) and off (made to work locally), turning on this option causes the terminal keypad to be turned on when `wgetch()` is called. The default value for keypad is false.

Initially, whether the terminal returns 7 or 8 significant bits on input depends on the control mode of the tty driver (see `termio(7I)`). To force 8 bits to be returned, invoke `meta(win, TRUE)`. To force 7 bits to be returned, invoke `meta(win, FALSE)`. The window argument, *win*, is always ignored. If the terminfo capabilities `smm` (`meta_on`) and `rmm` (`meta_off`) are defined for the terminal, `smm` is sent to the terminal when `meta(win, TRUE)` is called and `rmm` is sent when `meta(win, FALSE)` is called.

The `nodelay()` option causes `getch()` to be a non-blocking call. If no input is ready, `getch()` returns `ERR`. If disabled (*bf* is `FALSE`), `getch()` waits until a key is pressed.

While interpreting an input escape sequence, `wgetch()` sets a timer while waiting for the next character. If `notimeout(win, TRUE)` is called, then `wgetch()` does not set a timer. The purpose of the timeout is to differentiate between sequences received from a function key and those typed by a user.

With the `raw()` and `noraw()` routines, the terminal is placed into or out of raw mode. Raw mode is similar to `cbreak()` mode, in that characters typed are immediately passed through to the user program. The differences are that in raw mode, the interrupt, quit, suspend, and flow control characters are all passed through uninterpreted, instead of generating a signal. The behavior of the `BREAK` key depends on other bits in the tty driver that are not set by `curses`.

When the `noqiflush()` routine is used, normal flush of input and output queues associated with the `INTR`, `QUIT` and `SUSP` characters will not be done (see `termio(7I)`). When `qiflush()` is called, the queues will be flushed when these control characters are read.

The `timeout()` and `wtimeout()` routines set blocking or non-blocking read for a given window. If *delay* is negative, blocking read is used (that is, waits indefinitely for input). If *delay* is zero, then non-blocking read is used (that is, read returns `ERR` if no input is waiting). If *delay* is positive, then read blocks for *delay* milliseconds, and returns `ERR` if there is still no input. Hence, these routines provide the same functionality as `nodelay()`, plus the additional capability of being able to block for only *delay* milliseconds (where *delay* is positive).

`curses` does “line-breakout optimization” by looking for typeahead periodically while updating the screen. If input is found, and it is coming from a `tty`, the current update is postponed until `refresh()` or `doupdate()` is called again. This allows faster response to commands typed in advance. Normally, the input `FILE` pointer passed to `newterm()`, or `stdin` in the case that `initscr()` was used, will be used to do this typeahead checking. The `typeahead()` routine specifies that the file descriptor *fildev* is to be used to check for typeahead instead. If *fildev* is `-1`, then no typeahead checking is done.

RETURN VALUES All routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_getch(3CURSES)`, `curs_initscr(3CURSES)`, `curses(3CURSES)`, `attributes(5)`, `termio(7I)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `echo()`, `noecho()`, `halfdelay()`, `intrflush()`, `meta()`, `nodelay()`, `notimeout()`, `noqiflush()`, `qiflush()`, `timeout()`, and `wtimeout()` may be macros.

cbreak(3XCURSES)

| NAME | cbreak, nocbreak, noraw, raw – set input mode controls | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int cbreak(void); int nocbreak(void); int noraw(void); int raw(void);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>cbreak()</code> function enables the character input mode. This overrides any previous call to the <code>raw()</code> function and turns the <code>stty</code> flag <code>ICANON</code> off.</p> <p>The <code>nocbreak()</code> function sets the line canonical mode and turns the <code>stty</code> flag <code>ICANON</code> on without touching the <code>ISIG</code> or <code>IXON</code> flags.</p> <p>The <code>noraw()</code> function sets the line canonical mode and turns the the <code>stty</code> flags <code>ICANON</code>, <code>ISIG</code>, and <code>IXON</code> all on.</p> <p>The <code>raw()</code> function sets the character input mode and turns the <code>stty</code> flags <code>ICANON</code>, <code>ISIG</code>, and <code>IXON</code> all off. This mode provides maximum control over input.</p> <p>It is important to remember that the terminal may or may not be in character mode operation initially. Most interactive programs require <code>cbreak()</code> to be enabled.</p> | | | | | | |
| RETURN VALUES | On success, these functions return <code>OK</code> . Otherwise, they return <code>ERR</code> . | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Standard</td></tr><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>getch(3XCURSES)</code> , <code>halfdelay(3XCURSES)</code> , <code>nodelay(3XCURSES)</code> , <code>timeout(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> , <code>termio(7I)</code> | | | | | | |

| | | | | | | | | | | | | | | | |
|----------------------|---|----------|---|-------------|--|--------------|---|-------------|---|----------|---|----------|---|------------|---|
| NAME | chgat, mvchgat, mvwchgat, wchgat – change the rendition of characters in a window | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int chgat(int <i>n</i>, attr_t <i>attr</i>, short <i>color</i>, const void *<i>opts</i>); int mvchgat(int <i>y</i>, int <i>x</i>, int <i>n</i>, attr_t <i>attr</i>, short <i>color</i>, const void *<i>opts</i>); int mvwchgat(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>, int <i>n</i>, attr_t <i>attr</i>, short <i>color</i>, const void *<i>opts</i>); int wchgat(WINDOW *<i>win</i>, int <i>n</i>, attr_t <i>attr</i>, short <i>color</i>, const void *<i>opts</i>);</pre> | | | | | | | | | | | | | | |
| DESCRIPTION | <p>These functions change the renditions of the next <i>n</i> characters in the current or specified window (or of the remaining characters on the current or specified line, if <i>n</i> is -1), beginning at the current or specified cursor position. The attributes and colors are specified by <i>attr</i> and <i>color</i> as for <code>setcchar(3XCURSES)</code>.</p> <p>These function neither update the cursor nor perform wrapping.</p> <p>A value of <i>n</i> that is greater than the remaining characters on a line is not an error.</p> <p>The <i>opts</i> argument is reserved for definition in a future release. Currently, the application must provide a null pointer for <i>opts</i>.</p> | | | | | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>n</i></td> <td>Is the number of characters whose rendition is to be changed.</td> </tr> <tr> <td><i>attr</i></td> <td>Is the set of attributes to be assigned to the characters.</td> </tr> <tr> <td><i>color</i></td> <td>Is the new color pair to be assigned to the characters.</td> </tr> <tr> <td><i>opts</i></td> <td>Is reserved for future use. Currently, this must be a null pointer.</td> </tr> <tr> <td><i>y</i></td> <td>Is the y (row) coordinate of the starting position in the window.</td> </tr> <tr> <td><i>x</i></td> <td>Is the x (column) coordinate of the starting position in the window. changed in the window.</td> </tr> <tr> <td><i>win</i></td> <td>Is a pointer to the window in which the rendition of characters is to be changed.</td> </tr> </table> | <i>n</i> | Is the number of characters whose rendition is to be changed. | <i>attr</i> | Is the set of attributes to be assigned to the characters. | <i>color</i> | Is the new color pair to be assigned to the characters. | <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | <i>y</i> | Is the y (row) coordinate of the starting position in the window. | <i>x</i> | Is the x (column) coordinate of the starting position in the window. changed in the window. | <i>win</i> | Is a pointer to the window in which the rendition of characters is to be changed. |
| <i>n</i> | Is the number of characters whose rendition is to be changed. | | | | | | | | | | | | | | |
| <i>attr</i> | Is the set of attributes to be assigned to the characters. | | | | | | | | | | | | | | |
| <i>color</i> | Is the new color pair to be assigned to the characters. | | | | | | | | | | | | | | |
| <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | | | | | | | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of the starting position in the window. | | | | | | | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of the starting position in the window. changed in the window. | | | | | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the rendition of characters is to be changed. | | | | | | | | | | | | | | |
| RETURN VALUES | Upon successful completion, these functions returned OK. Otherwise, they return ERR. | | | | | | | | | | | | | | |
| ERRORS | No errors are defined. | | | | | | | | | | | | | | |

chgat(3XCURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `bkgrnd(3XCURSES)`, `libcurses(3XCURSES)`, `setcchar(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | plot, arc, box, circle, closepl, closevt, cont, erase, label, line, linemod, move, openpl, openvt, point, space – graphics interface |
| SYNOPSIS | <pre>cc [flag ...] file ... -lplot [library...] #include <plot.h> void arc(short x0, short y0, short x1, short y1, short x2, short y2); void box(short x0, short y0, short x1, short y1); void circle(short x, short y, short r); void closepl(); void closevt(); void cont(short x, short y); void erase(); void label(char *s); void line(short x0, short y0, short x1, short y1); void linemod(char *s); void move(short x, short y); void openpl(); void openvt(); void point(short x, short y); void space(short x0, short y0, short x1, short y1);</pre> |
| DESCRIPTION | <p>These functions generate graphics output for a set of output devices. The format of the output is dependent upon which link editor option is used when the program is compiled and linked (see Link Editor).</p> <p>The term "current point" refers to the current setting for the x and y coordinates.</p> <p>The <code>arc()</code> function specifies a circular arc. The coordinates $(x0, y0)$ specify the center of the arc. The coordinates $(x1, y1)$ specify the starting point of the arc. The coordinates $(x2, y2)$ specify the end point of the circular arc.</p> <p>The <code>box()</code> function specifies a rectangle with coordinates $(x0, y0)$, $(x0, y1)$, $(x1, y0)$, and $(x1, y1)$. The current point is set to $(x1, y1)$.</p> <p>The <code>circle()</code> function specifies a circle with a center at the coordinates (x, y) and a radius of r.</p> <p>The <code>closevt()</code> and <code>closepl()</code> functions flush the output.</p> <p>The <code>cont()</code> function specifies a line beginning at the current point and ending at the coordinates (x, y). The current point is set to (x, y).</p> |

circle(3PLOT)

The `erase()` function starts another frame of output.

The `label()` function places the null terminated string `s` so that the first character falls on the current point. The string is then terminated by a NEWLINE character.

The `line()` function draws a line starting at the coordinates (x_0, y_0) and ending at the coordinates (x_1, y_1) . The current point is set to (x_1, y_1) .

The `linemod()` function specifies the style for drawing future lines. `s` may contain one of the following: `dotted`, `solid`, `longdashed`, `shortdashed`, or `dotdashed`.

The `move()` function sets the current point to the coordinates (x, y) .

The `openpl()` or `openvt()` function must be called to open the device before any other plot functions are called.

The `point()` function plots the point given by the coordinates (x, y) . The current point is set to (x, y) .

The `space()` function specifies the size of the plotting area. The plot will be reduced or enlarged as necessary to fit the area specified. The coordinates (x_0, y_0) specify the lower left hand corner of the plotting area. The coordinates (x_1, y_1) specify the upper right hand corner of the plotting area.

Link Editor

Various flavors of these functions exist for different output devices. They are obtained by using the following `ld(1)` options:

| | |
|---------------------|--|
| <code>-lplot</code> | device-independent graphics stream on standard output in the format described in <code>plot(4B)</code> |
| <code>-l300</code> | GSI 300 terminal |
| <code>-l300s</code> | GSI 300S terminal |
| <code>-l4014</code> | Tektronix 4014 terminal |
| <code>-l450</code> | GSI 450 terminal |
| <code>-lvt0</code> | |

FILES

| | |
|--|----------------------|
| <code>/usr/lib/libplot.a</code> | archive library |
| <code>/usr/lib/libplot.so.1</code> | shared object |
| <code>/usr/lib/sparcv9/libplot.so.1</code> | 64-bit shared object |
| <code>/usr/lib/lib300.a</code> | archive library |
| <code>/usr/lib/lib300.so.1</code> | shared object |

```

/usr/lib/sparcv9/lib300.so.1
  64-bit shared object
/usr/lib/lib300s.a
  archive library
/usr/lib/lib300s.so.1
  shared object
/usr/lib/sparcv9/lib300s.so.1
  64-bit shared object
/usr/lib/lib4014.a
  archive library
/usr/lib/lib4014.so.1
  shared object
/usr/lib/sparcv9/lib4014.so.1
  64-bit shared object
/usr/lib/lib450.a
  archive library
/usr/lib/lib450.so.1
  shared object
/usr/lib/sparcv9/lib450.so.1
  64-bit shared object
/usr/lib/libvt0.a
  archive library
/usr/lib/libvt0.so.1
  shared object
/usr/lib/sparcv9/libvt0.so.1
  64-bit shared object

```

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `graph(1)`, `ld(1)`, `libplot(3LIB)`, `plot(4B)`, `attributes(5)`

clear(3CURSES)

| NAME | <code> curs_clear, erase, werase, clear, wclear, clrtoobot, wclrtoobot, clrtoeol, wclrtoeol </code> – clear all or part of a curses window | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file ... -lcurses [library ...] #include <curses.h> int erase(void); int werase(WINDOW *win); int clear(void); int wclear(WINDOW *win); int clrtoobot(void); int wclrtoobot(WINDOW *win); int clrtoeol(void); int wclrtoeol(WINDOW *win);</pre> | | | | |
| DESCRIPTION | <p>The <code>erase()</code> and <code>werase()</code> routines copy blanks to every position in the window.</p> <p>The <code>clear()</code> and <code>wclear()</code> routines are like <code>erase()</code> and <code>werase()</code>, but they also call <code>clearok()</code>, so that the screen is cleared completely on the next call to <code>wrefresh()</code> for that window and repainted from scratch.</p> <p>The <code>clrtoobot()</code> and <code>wclrtoobot()</code> routines erase all lines below the cursor in the window. Also, the current line to the right of the cursor, inclusive, is erased.</p> <p>The <code>clrtoeol()</code> and <code>wclrtoeol()</code> routines erase the current line to the right of the cursor, inclusive.</p> | | | | |
| RETURN VALUES | All routines return the integer <code>OK</code> , or a non-negative integer if <code>immedok()</code> is set. See <code>curls_ouptops(3CURSES)</code> . | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curls_ouptops(3CURSES)</code> , <code>curls_refresh(3CURSES)</code> , <code>curls(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that <code>erase()</code>, <code>werase()</code>, <code>clear()</code>, <code>wclear()</code>, <code>clrtoobot()</code>, and <code>clrtoeol()</code> may be macros.</p> | | | | |

| NAME | clear, erase, wclear, werase – clear a window | | | | | | |
|---------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int clear(void); int erase(void); int wclear(WINDOW *win); int werase(WINDOW *win);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>clear()</code> and <code>erase()</code> functions clear <code>stdscr</code>, destroying its previous contents. The <code>wclear()</code> and <code>werase()</code> functions perform the same action, but clear the window specified by <code>win</code> instead of <code>stdscr</code>.</p> <p>The <code>clear()</code> and <code>wclear()</code> functions also call the <code>clearok()</code> function. This function clears and redraws the entire screen on the next call to <code>refresh(3XCURSES)</code> or <code>wrefresh(3XCURSES)</code> for the window.</p> <p>The current background character (and attributes) is used to clear the screen.</p> | | | | | | |
| PARAMETERS | <p><i>win</i> Is a pointer to the window that is to be cleared.</p> | | | | | | |
| ERRORS | <p>OK Successful completion.</p> <p>ERR An error occurred.</p> | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <p><code>bkgdset(3XCURSES)</code>, <code>clearok(3XCURSES)</code>, <code>clrtoeol(3XCURSES)</code>, <code>clrtoeol(3XCURSES)</code>, <code>doupdate(3XCURSES)</code>, <code>libcurses(3XCURSES)</code>, <code>refresh(3XCURSES)</code>, <code>wrefresh(3XCURSES)</code>, <code>attributes(5)</code>, <code>standards(5)</code></p> | | | | | | |

clearok(3CURSES)

| | |
|--------------------|--|
| NAME | <code>curs_ouopts</code> , <code>clearok</code> , <code>idlok</code> , <code>idcok</code> , <code>immedok</code> , <code>leaveok</code> , <code>setscrreg</code> , <code>wsetscrreg</code> , <code>scrollok</code> , <code>nl</code> , <code>nonl</code> – curses terminal output option control routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int clearok(WINDOW *win, bool bf); int idlok(WINDOW *win, bool bf); void idcok(WINDOW *win, bool bf); void immedok(WINDOW *win, bool bf); int leaveok(WINDOW *win, bool bf); int setscrreg(int top, int bot); int wsetscrreg(WINDOW *win, int top, int bot); int scrollok(WINDOW *win, bool bf); int nl(void); int nonl(void);</pre> |
| DESCRIPTION | <p>These routines set options that deal with output within curses. All options are initially FALSE, unless otherwise stated. It is not necessary to turn these options off before calling <code>endwin()</code>.</p> <p>With the <code>clearok()</code> routine, if enabled (<i>bf</i> is TRUE), the next call to <code>wrefresh()</code> with this window will clear the screen completely and redraw the entire screen from scratch. This is useful when the contents of the screen are uncertain, or in some cases for a more pleasing visual effect. If the <i>win</i> argument to <code>clearok()</code> is the global variable <code>curscr()</code>, the next call to <code>wrefresh()</code> with any window causes the screen to be cleared and repainted from scratch.</p> <p>With the <code>idlok()</code> routine, if enabled (<i>bf</i> is TRUE), curses considers using the hardware insert/delete line feature of terminals so equipped. If disabled (<i>bf</i> is FALSE), curses very seldom uses this feature. (The insert/delete character feature is always considered.) This option should be enabled only if the application needs insert/delete line, for example, for a screen editor. It is disabled by default because insert/delete line tends to be visually annoying when used in applications where it isn't really needed. If insert/delete line cannot be used, curses redraws the changed portions of all lines.</p> <p>With the <code>idcok()</code> routine, if enabled (<i>bf</i> is TRUE), curses considers using the hardware insert/delete character feature of terminals so equipped. This is enabled by default.</p> <p>With the <code>immedok()</code> routine, if enabled (<i>bf</i> is TRUE), any change in the window image, such as the ones caused by <code>waddch()</code>, <code>wclrtoeol()</code>, <code>wscrl()</code>, etc., automatically cause a call to <code>wrefresh()</code>. However, it may degrade the performance considerably, due to repeated calls to <code>wrefresh()</code>. It is disabled by default.</p> |

clearok(3CURSES)

Normally, the hardware cursor is left at the location of the window cursor being refreshed. The `leaveok()` option allows the cursor to be left wherever the update happens to leave it. It is useful for applications where the cursor is not used, since it reduces the need for cursor motions. If possible, the cursor is made invisible when this option is enabled.

The `setscrreg()` and `wsetscrreg()` routines allow the application programmer to set a software scrolling region in a window. *top* and *bot* are the line numbers of the top and bottom margin of the scrolling region. (Line 0 is the top line of the window.) If this option and `scrollok()` are enabled, an attempt to move off the bottom margin line causes all lines in the scrolling region to scroll up one line. Only the text of the window is scrolled. (Note that this has nothing to do with the use of a physical scrolling region capability in the terminal, like that in the VT100. If `idlok()` is enabled and the terminal has either a scrolling region or insert/delete line capability, they will probably be used by the output routines.)

The `scrollok()` option controls what happens when the cursor of a window is moved off the edge of the window or scrolling region, either as a result of a newline action on the bottom line, or typing the last character of the last line. If disabled, (*bf* is FALSE), the cursor is left on the bottom line. If enabled, (*bf* is TRUE), `wrefresh()` is called on the window, and the physical terminal and window are scrolled up one line. (Note that in order to get the physical scrolling effect on the terminal, it is also necessary to call `idlok()`.)

The `nl()` and `nonl()` routines control whether newline is translated into carriage return and linefeed on output, and whether return is translated into newline on input. Initially, the translations do occur. By disabling these translations using `nonl()`, `curses` is able to make better use of the linefeed capability, resulting in faster cursor motion.

RETURN VALUES `setscrreg()` and `wsetscrreg()` return OK upon success and ERR upon failure. All other routines that return an integer always return OK.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_addch(3CURSES)`, `curs_clear(3CURSES)`, `curs_initscr(3CURSES)`, `curs_refresh(3CURSES)`, `curs_scroll(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `clearok()`, `leaveok()`, `scrollok()`, `idcok()`, `nl()`, `nonl()`, and `setscrreg()` may be macros.

`clearok(3CURSES)`

The `immedok()` routine is useful for windows that are used as terminal emulators.

| | |
|--------------------|---|
| NAME | clearok, idlok, leaveok, scrollok, setscrreg, wsetscreg – terminal output control functions |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int clearok(WINDOW *win, bool bf); int idlok(WINDOW *win, bool bf); int leaveok(WINDOW *win, bool bf); int scrollok(WINDOW *win, bool bf); int setscreg(int top, int bot); int wsetscreg(WINDOW *win, int top, int bot);</pre> |
| DESCRIPTION | <p>These functions set options that deal with the output within Curses functions.</p> <p>The <code>clearok()</code> function assigns the value of <i>bf</i> to an internal flag in the specified window that governs clearing of the screen during a refresh. If, during a refresh operation on the specified window, the flag in <i>curscr</i> is TRUE or the flag in the specified window is TRUE, <code>clearok()</code> clears the screen, redraws it in its entirety, and sets the flag to FALSE in <i>curscr</i> and in the specified window. The initial state is unspecified</p> <p>The <code>idlok()</code> function specifies whether the implementation may use the hardware insert-line, delete-line, and scroll features of terminals so equipped. If <i>bf</i> is TRUE, use of these features is enabled. If <i>bf</i> is FALSE, use of these features is disabled and lines are instead redrawn as required. The initial state is FALSE.</p> <p>The <code>leaveok()</code> function controls the cursor position after a refresh operation. If <i>bf</i> is TRUE, refresh operations on the specified window may leave the terminal's cursor at an arbitrary position. If <i>bf</i> is FALSE, then at the end of any refresh operation, the terminal's cursor is positioned at the cursor position contained in the specified window. The initial state is FALSE.</p> <p>The <code>scrollok()</code> function controls the use of scrolling. If <i>bf</i> is TRUE, then scrolling is enabled for the specified window. If <i>bf</i> is FALSE, scrolling is disabled for the specified window. The initial state is FALSE.</p> <p>The <code>setscreg()</code> and <code>wsetscreg()</code> functions define a software scrolling region in the current or specified window. The <i>top</i> and <i>bottom</i> arguments are the line numbers of the first and last line defining the scrolling region. (Line 0 is the top line of the window.) If this option and <code>scrollok()</code> are enabled, an attempt to move off the last line of the margin causes all lines in the scrolling region to scroll one line in the</p> |

clearok(3XCURSES)

direction of the first line. Only characters in the window are scrolled. If a software scrolling region is set and `scrollok()` is not enabled, an attempt to move off the last line of the margin does not reposition any lines in the scrolling region.

PARAMETERS

win Is a pointer to a window.
bf Is a Boolean expression.
top Is the top line of the scrolling region (top of the window is line 0).
bot Is the bottom line of the scrolling region (top of the window is line 0).

RETURN VALUES

Upon successful completion, the `setscrreg()` and `wsetscrreg()` functions return OK. Otherwise, they return ERR.

The other functions always return OK.

ERRORS

No errors are defined.

USAGE

The only reason to enable the `idlok()` feature is to use scrolling to achieve the visual effect of motion of a partial window, such as for a screen editor. In other cases, the feature can be visually annoying.

The `leaveok()` option provides greater efficiency for applications that do not use the cursor.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO

`bkgdset(3XCURSES)`, `clear(3XCURSES)`, `doupdate(3XCURSES)`, `libcurses(3XCURSES)`, `scr1(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | plot, arc, box, circle, closepl, closevt, cont, erase, label, line, linemod, move, openpl, openvt, point, space – graphics interface |
| SYNOPSIS | <pre>cc [flag ...] file ... -lplot [library...] #include <plot.h> void arc(short x0, short y0, short x1, short y1, short x2, short y2); void box(short x0, short y0, short x1, short y1); void circle(short x, short y, short r); void closepl(); void closevt(); void cont(short x, short y); void erase(); void label(char *s); void line(short x0, short y0, short x1, short y1); void linemod(char *s); void move(short x, short y); void openpl(); void openvt(); void point(short x, short y); void space(short x0, short y0, short x1, short y1);</pre> |
| DESCRIPTION | <p>These functions generate graphics output for a set of output devices. The format of the output is dependent upon which link editor option is used when the program is compiled and linked (see Link Editor).</p> <p>The term "current point" refers to the current setting for the x and y coordinates.</p> <p>The <code>arc()</code> function specifies a circular arc. The coordinates $(x0, y0)$ specify the center of the arc. The coordinates $(x1, y1)$ specify the starting point of the arc. The coordinates $(x2, y2)$ specify the end point of the circular arc.</p> <p>The <code>box()</code> function specifies a rectangle with coordinates $(x0, y0)$, $(x0, y1)$, $(x1, y0)$, and $(x1, y1)$. The current point is set to $(x1, y1)$.</p> <p>The <code>circle()</code> function specifies a circle with a center at the coordinates (x, y) and a radius of r.</p> <p>The <code>closevt()</code> and <code>closepl()</code> functions flush the output.</p> <p>The <code>cont()</code> function specifies a line beginning at the current point and ending at the coordinates (x, y). The current point is set to (x, y).</p> |

closepl(3PLOT)

The `erase()` function starts another frame of output.

The `label()` function places the null terminated string `s` so that the first character falls on the current point. The string is then terminated by a NEWLINE character.

The `line()` function draws a line starting at the coordinates (x_0, y_0) and ending at the coordinates (x_1, y_1) . The current point is set to (x_1, y_1) .

The `linemod()` function specifies the style for drawing future lines. `s` may contain one of the following: `dotted`, `solid`, `longdashed`, `shortdashed`, or `dotdashed`.

The `move()` function sets the current point to the coordinates (x, y) .

The `openpl()` or `openvt()` function must be called to open the device before any other plot functions are called.

The `point()` function plots the point given by the coordinates (x, y) . The current point is set to (x, y) .

The `space()` function specifies the size of the plotting area. The plot will be reduced or enlarged as necessary to fit the area specified. The coordinates (x_0, y_0) specify the lower left hand corner of the plotting area. The coordinates (x_1, y_1) specify the upper right hand corner of the plotting area.

Link Editor Various flavors of these functions exist for different output devices. They are obtained by using the following `ld(1)` options:

| | |
|---------------------|--|
| <code>-lplot</code> | device-independent graphics stream on standard output in the format described in <code>plot(4B)</code> |
| <code>-l300</code> | GSI 300 terminal |
| <code>-l300s</code> | GSI 300S terminal |
| <code>-l4014</code> | Tektronix 4014 terminal |
| <code>-l450</code> | GSI 450 terminal |
| <code>-lvt0</code> | |

FILES

| | |
|--|----------------------|
| <code>/usr/lib/libplot.a</code> | archive library |
| <code>/usr/lib/libplot.so.1</code> | shared object |
| <code>/usr/lib/sparcv9/libplot.so.1</code> | 64-bit shared object |
| <code>/usr/lib/lib300.a</code> | archive library |
| <code>/usr/lib/lib300.so.1</code> | shared object |

```

/usr/lib/sparcv9/lib300.so.1
  64-bit shared object
/usr/lib/lib300s.a
  archive library
/usr/lib/lib300s.so.1
  shared object
/usr/lib/sparcv9/lib300s.so.1
  64-bit shared object
/usr/lib/lib4014.a
  archive library
/usr/lib/lib4014.so.1
  shared object
/usr/lib/sparcv9/lib4014.so.1
  64-bit shared object
/usr/lib/lib450.a
  archive library
/usr/lib/lib450.so.1
  shared object
/usr/lib/sparcv9/lib450.so.1
  64-bit shared object
/usr/lib/libvt0.a
  archive library
/usr/lib/libvt0.so.1
  shared object
/usr/lib/sparcv9/libvt0.so.1
  64-bit shared object

```

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `graph(1)`, `ld(1)`, `libplot(3LIB)`, `plot(4B)`, `attributes(5)`

closevt(3PLOT)

| | |
|--------------------|--|
| NAME | plot, arc, box, circle, closepl, closevt, cont, erase, label, line, linemod, move, openpl, openvt, point, space – graphics interface |
| SYNOPSIS | <pre>cc [flag ...] file ... -lplot [library...] #include <plot.h> void arc(short x0, short y0, short x1, short y1, short x2, short y2); void box(short x0, short y0, short x1, short y1); void circle(short x, short y, short r); void closepl(); void closevt(); void cont(short x, short y); void erase(); void label(char *s); void line(short x0, short y0, short x1, short y1); void linemod(char *s); void move(short x, short y); void openpl(); void openvt(); void point(short x, short y); void space(short x0, short y0, short x1, short y1);</pre> |
| DESCRIPTION | <p>These functions generate graphics output for a set of output devices. The format of the output is dependent upon which link editor option is used when the program is compiled and linked (see Link Editor).</p> <p>The term "current point" refers to the current setting for the x and y coordinates.</p> <p>The <code>arc()</code> function specifies a circular arc. The coordinates $(x0, y0)$ specify the center of the arc. The coordinates $(x1, y1)$ specify the starting point of the arc. The coordinates $(x2, y2)$ specify the end point of the circular arc.</p> <p>The <code>box()</code> function specifies a rectangle with coordinates $(x0, y0)$, $(x0, y1)$, $(x1, y0)$, and $(x1, y1)$. The current point is set to $(x1, y1)$.</p> <p>The <code>circle()</code> function specifies a circle with a center at the coordinates (x, y) and a radius of r.</p> <p>The <code>closevt()</code> and <code>closepl()</code> functions flush the output.</p> <p>The <code>cont()</code> function specifies a line beginning at the current point and ending at the coordinates (x, y). The current point is set to (x, y).</p> |

The `erase()` function starts another frame of output.

The `label()` function places the null terminated string `s` so that the first character falls on the current point. The string is then terminated by a NEWLINE character.

The `line()` function draws a line starting at the coordinates $(x0, y0)$ and ending at the coordinates $(x1, y1)$. The current point is set to $(x1, y1)$.

The `linemod()` function specifies the style for drawing future lines. `s` may contain one of the following: `dotted`, `solid`, `longdashed`, `shortdashed`, or `dotdashed`.

The `move()` function sets the current point to the coordinates (x, y) .

The `openpl()` or `openvt()` function must be called to open the device before any other plot functions are called.

The `point()` function plots the point given by the coordinates (x, y) . The current point is set to (x, y) .

The `space()` function specifies the size of the plotting area. The plot will be reduced or enlarged as necessary to fit the area specified. The coordinates $(x0, y0)$ specify the lower left hand corner of the plotting area. The coordinates $(x1, y1)$ specify the upper right hand corner of the plotting area.

Link Editor Various flavors of these functions exist for different output devices. They are obtained by using the following `ld(1)` options:

```
-lplot          device-independent graphics stream on standard output in the
                 format described in plot(4B)
-l300           GSI 300 terminal
-l300s          GSI 300S terminal
-l4014          Tektronix 4014 terminal
-l450           GSI 450 terminal
-lvt0
```

FILES

```
/usr/lib/libplot.a
  archive library
/usr/lib/libplot.so.1
  shared object
/usr/lib/sparcv9/libplot.so.1
  64-bit shared object
/usr/lib/lib300.a
  archive library
/usr/lib/lib300.so.1
  shared object
```

closevt(3PLOT)

/usr/lib/sparcv9/lib300.so.1
64-bit shared object

/usr/lib/lib300s.a
archive library

/usr/lib/lib300s.so.1
shared object

/usr/lib/sparcv9/lib300s.so.1
64-bit shared object

/usr/lib/lib4014.a
archive library

/usr/lib/lib4014.so.1
shared object

/usr/lib/sparcv9/lib4014.so.1
64-bit shared object

/usr/lib/lib450.a
archive library

/usr/lib/lib450.so.1
shared object

/usr/lib/sparcv9/lib450.so.1
64-bit shared object

/usr/lib/libvt0.a
archive library

/usr/lib/libvt0.so.1
shared object

/usr/lib/sparcv9/libvt0.so.1
64-bit shared object

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO graph(1), ld(1), libplot(3LIB), plot(4B), attributes(5)

NAME | curs_clear, erase, werase, clear, wclear, clrrobot, wclrrobot, clrtoeol, wclrtoeol – clear all or part of a curses window

SYNOPSIS | `cc [flag...] file ... -lcurses [library ...]
#include <curses.h>`

```
int erase(void);
int werase(WINDOW *win);
int clear(void);
int wclear(WINDOW *win);
int clrrobot(void);
int wclrrobot(WINDOW *win);
int clrtoeol(void);
int wclrtoeol(WINDOW *win);
```

DESCRIPTION | The `erase()` and `werase()` routines copy blanks to every position in the window.

The `clear()` and `wclear()` routines are like `erase()` and `werase()`, but they also call `clearok()`, so that the screen is cleared completely on the next call to `wrefresh()` for that window and repainted from scratch.

The `clrrobot()` and `wclrrobot()` routines erase all lines below the cursor in the window. Also, the current line to the right of the cursor, inclusive, is erased.

The `clrtoeol()` and `wclrtoeol()` routines erase the current line to the right of the cursor, inclusive.

RETURN VALUES | All routines return the integer OK, or a non-negative integer if `immedok()` is set. See `curs_ouptos(3CURSES)`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | `curs_ouptos(3CURSES)`, `curs_refresh(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES | The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `erase()`, `werase()`, `clear()`, `wclear()`, `clrrobot()`, and `clrtoeol()` may be macros.

clrrobot(3XCURSES)

| NAME | clrrobot, wclrrobot – clear to the end of a window | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int clrrobot(void); int wclrrobot(WINDOW *win);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>clrrobot()</code> function clears all characters in the <code>stdscr</code> window from the cursor to the end of the window. The <code>wclrrobot()</code> function performs the same action in the window specified by <i>win</i> instead of in <code>stdscr</code>. The current background character (and rendition) is used to clear the screen.</p> <p>If the clearing action results in clearing only a portion of a multicolumn character, background characters are displayed in place of the remaining portion.</p> | | | | | | |
| PARAMETERS | <i>win</i> Is a pointer to the window that is to be cleared. | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Standard</td></tr><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>bkgdset(3XCURSES)</code> , <code>clear(3XCURSES)</code> , <code>clearok(3XCURSES)</code> , <code>crltoeol(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

| NAME | curs_clear, erase, werase, clear, wclear, clrtoeol, wclrtoeol, clrrobot, wclrrobot, clrtoeol, wclrtoeol – clear all or part of a curses window | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file ... -lcurses [library ...] #include <curses.h> int erase(void); int werase(WINDOW *win); int clear(void); int wclear(WINDOW *win); int clrrobot(void); int wclrrobot(WINDOW *win); int clrtoeol(void); int wclrtoeol(WINDOW *win);</pre> | | | | |
| DESCRIPTION | <p>The erase() and werase() routines copy blanks to every position in the window.</p> <p>The clear() and wclear() routines are like erase() and werase(), but they also call clearok(), so that the screen is cleared completely on the next call to wrefresh() for that window and repainted from scratch.</p> <p>The clrrobot() and wclrrobot() routines erase all lines below the cursor in the window. Also, the current line to the right of the cursor, inclusive, is erased.</p> <p>The clrtoeol() and wclrtoeol() routines erase the current line to the right of the cursor, inclusive.</p> | | | | |
| RETURN VALUES | All routines return the integer OK, or a non-negative integer if immedok() is set. See curs_ouptos(3CURSES). | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curs_ouptos(3CURSES), curs_refresh(3CURSES), curses(3CURSES), attributes(5) | | | | |
| NOTES | <p>The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.</p> <p>Note that erase(), werase(), clear(), wclear(), clrrobot(), and clrtoeol() may be macros.</p> | | | | |

clrtoeol(3XCURSES)

| NAME | clrtoeol, wclrtoeol – clear to the end of a line | | | | | | |
|----------------------|--|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int clrtoeol(void); int wclrtoeol(WINDOW *win);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>clrtoeol()</code> function clears the current line from the cursor to the right margin in the <code>stdscr</code> window. The <code>wclrtoeol()</code> function performs the same action, but in the window specified by <code>win</code> instead of <code>stdscr</code>. The current background character (and rendition) is used to clear the screen.</p> <p>If the clearing action results in clearing only a portion of a multicolumn character, background characters are displayed in place of the remaining portion.</p> | | | | | | |
| PARAMETERS | <p><i>win</i> Is a pointer to the window in which to clear to the end of the line.</p> | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return FALSE. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Standard</td></tr><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>bkgdset(3XCURSES)</code> , <code>clear(3XCURSES)</code> , <code>clearok(3XCURSES)</code> , <code>clrtoeol(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

| | |
|-----------------------------|--|
| NAME | curs_color, start_color, init_pair, init_color, has_colors, can_change_color, color_content, pair_content – curses color manipulation routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int start_color(void); int init_pair(short pair, short fg, short bg); int init_color(short color, short red, short green, short blue); bool has_colors(void); bool can_change_color(void); int color_content(short color, short *redp, short *greenp, short *bluep); int pair_content(short pair, short *fgp, short *bgp);</pre> |
| Overview | <p>curses provides routines that manipulate color on color alphanumeric terminals. To use these routines <code>start_color()</code> must be called, usually right after <code>initscr()</code>. See <code>curs_initscr(3CURSES)</code>. Colors are always used in pairs (referred to as color-pairs). A color-pair consists of a foreground color (for characters) and a background color (for the field on which the characters are displayed). A programmer initializes a color-pair with the routine <code>init_pair</code>. After it has been initialized, <code>COLOR_PAIR(n)</code>, a macro defined in <code><curses.h></code>, can be used in the same ways other video attributes can be used. If a terminal is capable of redefining colors, the programmer can use the routine <code>init_color()</code> to change the definition of a color. The routines <code>has_colors()</code> and <code>can_change_color()</code> return <code>TRUE</code> or <code>FALSE</code>, depending on whether the terminal has color capabilities and whether the programmer can change the colors. The routine <code>color_content()</code> allows a programmer to identify the amounts of red, green, and blue components in an initialized color. The routine <code>pair_content()</code> allows a programmer to find out how a given color-pair is currently defined.</p> |
| Routine Descriptions | <p>The <code>start_color()</code> routine requires no arguments. It must be called if the programmer wants to use colors, and before any other color manipulation routine is called. It is good practice to call this routine right after <code>initscr()</code>. <code>start_color()</code> initializes eight basic colors (black, red, green, yellow, blue, magenta, cyan, and white), and two global variables, <code>COLORS</code> and <code>COLOR_PAIRS</code> (respectively defining the maximum number of colors and color-pairs the terminal can support). It also restores the colors on the terminal to the values they had when the terminal was just turned on.</p> <p>The <code>init_pair()</code> routine changes the definition of a color-pair. It takes three arguments: the number of the color-pair to be changed, the foreground color number, and the background color number. The value of the first argument must be between 1 and <code>COLOR_PAIRS-1</code>. The value of the second and third arguments must be between 0 and <code>COLORS</code>. If the color-pair was previously initialized, the screen is refreshed and all occurrences of that color-pair is changed to the new definition.</p> |

color_content(3CURSES)

The `init_color()` routine changes the definition of a color. It takes four arguments: the number of the color to be changed followed by three RGB values (for the amounts of red, green, and blue components). The value of the first argument must be between 0 and `COLORS`. (See the section `COLORS` for the default color index.) Each of the last three arguments must be a value between 0 and 1000. When `init_color()` is used, all occurrences of that color on the screen immediately change to the new definition.

The `has_colors()` routine requires no arguments. It returns `TRUE` if the terminal can manipulate colors; otherwise, it returns `FALSE`. This routine facilitates writing terminal-independent programs. For example, a programmer can use it to decide whether to use color or some other video attribute.

The `can_change_color()` routine requires no arguments. It returns `TRUE` if the terminal supports colors and can change their definitions; other, it returns `FALSE`. This routine facilitates writing terminal-independent programs.

The `color_content()` routine gives users a way to find the intensity of the red, green, and blue (RGB) components in a color. It requires four arguments: the color number, and three addresses of `shorts` for storing the information about the amounts of red, green, and blue components in the given color. The value of the first argument must be between 0 and `COLORS`. The values that are stored at the addresses pointed to by the last three arguments are between 0 (no component) and 1000 (maximum amount of component).

The `pair_content()` routine allows users to find out what colors a given color-pair consists of. It requires three arguments: the color-pair number, and two addresses of `shorts` for storing the foreground and the background color numbers. The value of the first argument must be between 1 and `COLOR_PAIRS-1`. The values that are stored at the addresses pointed to by the second and third arguments are between 0 and `COLORS`.

Colors In `< curses.h >` the following macros are defined. These are the default colors. `curses` also assumes that `COLOR_BLACK` is the default background color for all terminals.

```
COLOR_BLACK
COLOR_RED
COLOR_GREEN
COLOR_YELLOW
COLOR_BLUE
COLOR_MAGENTA
COLOR_CYAN
COLOR_WHITE
```

RETURN VALUES All routines that return an integer return `ERR` upon failure and `OK` upon successful completion.

color_content(3CURSES)

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_attr(3CURSES), curs_initscr(3CURSES), curses(3CURSES), attributes(5)

NOTES The header < curses.h > automatically includes the headers < stdio.h > and < unctrl.h >.

color_content(3XCURSES)

| | |
|------------------------------|---|
| NAME | <code>can_change_color</code> , <code>color_content</code> , <code>COLOR_PAIR</code> , <code>has_colors</code> , <code>init_color</code> , <code>init_pair</code> , <code>pair_content</code> , <code>PAIR_NUMBER</code> , <code>start_color</code> , <code>COLOR_PAIRS</code> , <code>COLORS</code> – manipulate color information |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> bool can_change_color(void); int color_content(short color, short *red, short *green, short *blue); int COLOR_PAIR(int n); bool has_colors(void); int init_color(short color, short red, short green, short blue); int init_pair(short pair, short f, short b); int pair_content(short pair, short *f, short *b); int PAIR_NUMBER(int value); int start_color(void); extern int COLOR_PAIRS; extern int COLORS;</pre> |
| DESCRIPTION | These functions manipulate color on terminals that support color. |
| Querying Capabilities | The <code>has_colors()</code> function indicates whether the terminal is a color terminal. The <code>can_change_color()</code> function indicates whether the terminal is a color terminal on which colors can be redefined. |
| Initialization | <p>The <code>start_color()</code> function must be called to enable use of colors and before any color manipulation function is called. The function initializes eight basic colors (black, red, green, yellow, blue, magenta, cyan, and white) that can be specified by the color macros (such as <code>COLOR_BLACK</code>) defined in <code><curses.h></code>. The initial appearance of these colors is unspecified.</p> <p>The function also initializes two global external variables:</p> <ul style="list-style-type: none">■ <code>COLORS</code> defines the number of colors that the terminal supports. See Color Identification below. If <code>COLORS</code> is 0, the terminal does not support redefinition of colors and <code>can_change_color()</code> will return <code>FALSE</code>.■ <code>COLOR_PAIRS</code> defines the maximum number of color-pairs that the terminal supports. See User-defined Color Pairs below. |

color_content(3XCURSES)

Color Identification

The `start_color()` function also restores the colors on the terminal to terminal-specific initial values. The initial background color is assumed to be black for all terminals.

The `init_color()` function redefines color number *color*, on terminals that support the redefinition of colors, to have the red, green, and blue intensity components specified by *red*, *green*, and *blue*, respectively. Calling `init_color()` also changes all occurrences of the specified color on the screen to the new definition.

The `color_content()` function identifies the intensity components of color number *color*. It stores the red, green, and blue intensity components of this color in the addresses pointed to by *red*, *green*, and *blue*, respectively.

For both functions, the *color* argument must be in the range from 0 to and including `COLORS-1`. Valid intensity value range from 0 (no intensity component) up to and including 1000 (maximum intensity in that component).

User-defined Color Pairs

Calling `init_pair()` defines or redefines color-pair number *pair* to have foreground color *f* and background color *b*. Calling `init_pair()` changes any characters that were displayed in the color pair's old definition to the new definition and refreshes the screen.

After defining the color pair, the macro `COLOR_PAIR(n)` returns the value of color pair *n*. This value is the color attribute as it would be extracted from a `chtype`. Controversy, the macro `COLOR_NUMBER(value)` returns the color pair number associated with the color attribute *value*.

The `pair_content()` retrieves the component colors of a color-pair number *pair*. It stores the foreground and background color numbers in the variables pointed to by *f* and *b*, respectively.

With `init_pair()` and `pair_content()`, the value of *pair* must be in a range from 0 to and including `COLOR_PAIRS-1`. Valid values for *f* and *b* are the range from 0 to and including `COLORS-1`.

PARAMETERS

| | |
|--------------|--|
| <i>color</i> | Is the number of the color for which to provide information (0 to <code>COLORS-1</code>). |
| <i>red</i> | Is a pointer to the RGB value for the amount of red in <i>color</i> . |
| <i>green</i> | Is a pointer to the RGB value for the amount of green in <i>color</i> . |
| <i>blue</i> | Is a pointer to the RGB value for the amount of blue in <i>color</i> . |
| <i>n</i> | Is the number of a color pair. |
| <i>pair</i> | Is the number of the color pair for which to provide information (1 to <code>COLOR_PAIRS-1</code>). |
| <i>f</i> | Is a pointer to the number of the foreground color (0 to <code>COLORS-1</code>) in <i>pair</i> . |

color_content(3XCURSES)

b Is a pointer to the number of the background color (0 to COLORS-1) in *pair*.

value Is a color attribute value.

RETURN VALUES The `has_colors()` function returns `TRUE` if the terminal can manipulate colors. Otherwise, it returns `FALSE`.

The `can_change_color()` function returns `TRUE` if the terminal supports colors and is able to change their definitions. Otherwise, it returns `FALSE`.

Upon successful completion, the other functions return `OK`. Otherwise, they return `ERR`.

ERRORS No errors are defined.

USAGE To use these functions, `start_color()` must be called, usually right after `initscr(3XCURSES)`.

The `can_change_color()` and `has_colors()` functions facilitate writing terminal-independent applications. For example, a programmer can use them to decide whether to use color or some other video attribute.

On color terminals, a typical value of `COLORS` is 8 and the macros such as `COLOR_BLACK` return a value within the range from 0 to and including 7. However, applications cannot rely on this to be true.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attroff(3XCURSES)`, `delscreen(3XCURSES)`, `initscr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

COLOR_PAIR(3XCURSES)

| | |
|------------------------------|--|
| NAME | can_change_color, color_content, COLOR_PAIR, has_colors, init_color, init_pair, pair_content, PAIR_NUMBER, start_color, COLOR_PAIRS, COLORS – manipulate color information |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> bool can_change_color(void); int color_content(short color, short *red, short *green, short *blue); int COLOR_PAIR(int n); bool has_colors(void); int init_color(short color, short red, short green, short blue); int init_pair(short pair, short f, short b); int pair_content(short pair, short *f, short *b); int PAIR_NUMBER(int value); int start_color(void); extern int COLOR_PAIRS; extern int COLORS;</pre> |
| DESCRIPTION | These functions manipulate color on terminals that support color. |
| Querying Capabilities | The has_colors() function indicates whether the terminal is a color terminal. The can_change_color() function indicates whether the terminal is a color terminal on which colors can be redefined. |
| Initialization | <p>The start_color() function must be called to enable use of colors and before any color manipulation function is called. The function initializes eight basic colors (black, red, green, yellow, blue, magenta, cyan, and white) that can be specified by the color macros (such as COLOR_BLACK) defined in <curses.h>. The initial appearance of these colors is unspecified.</p> <p>The function also initializes two global external variables:</p> <ul style="list-style-type: none"> ■ COLORS defines the number of colors that the terminal supports. See Color Identification below. If COLORS is 0, the terminal does not support redefinition of colors and can_change_color() will return FALSE. ■ COLOR_PAIRS defines the maximum number of color-pairs that the terminal supports. See User-defined Color Pairs below. |

COLOR_PAIR(3XCURSES)

| | | | | | | | | | | | | | | | |
|---------------------------------|--|--------------|--|------------|---|--------------|---|-------------|--|----------|--------------------------------|-------------|--|----------|---|
| | <p>The <code>start_color()</code> function also restores the colors on the terminal to terminal-specific initial values. The initial background color is assumed to be black for all terminals.</p> | | | | | | | | | | | | | | |
| Color Identification | <p>The <code>init_color()</code> function redefines color number <i>color</i>, on terminals that support the redefinition of colors, to have the red, green, and blue intensity components specified by <i>red</i>, <i>green</i>, and <i>blue</i>, respectively. Calling <code>init_color()</code> also changes all occurrences of the specified color on the screen to the new definition.</p> <p>The <code>color_content()</code> function identifies the intensity components of color number <i>color</i>. It stores the red, green, and blue intensity components of this color in the addresses pointed to by <i>red</i>, <i>green</i>, and <i>blue</i>, respectively.</p> <p>For both functions, the <i>color</i> argument must be in the range from 0 to and including <code>COLORS-1</code>. Valid intensity value range from 0 (no intensity component) up to and including 1000 (maximum intensity in that component).</p> | | | | | | | | | | | | | | |
| User-defined Color Pairs | <p>Calling <code>init_pair()</code> defines or redefines color-pair number <i>pair</i> to have foreground color <i>f</i> and background color <i>b</i>. Calling <code>init_pair()</code> changes any characters that were displayed in the color pair's old definition to the new definition and refreshes the screen.</p> <p>After defining the color pair, the macro <code>COLOR_PAIR(<i>n</i>)</code> returns the value of color pair <i>n</i>. This value is the color attribute as it would be extracted from a <code>chtype</code>. Controversy, the macro <code>COLOR_NUMBER(<i>value</i>)</code> returns the color pair number associated with the color attribute <i>value</i>.</p> <p>The <code>pair_content()</code> retrieves the component colors of a color-pair number <i>pair</i>. It stores the foreground and background color numbers in the variables pointed to by <i>f</i> and <i>b</i>, respectively.</p> <p>With <code>init_pair()</code> and <code>pair_content()</code>, the value of <i>pair</i> must be in a range from 0 to and including <code>COLOR_PAIRS-1</code>. Valid values for <i>f</i> and <i>b</i> are the range from 0 to and including <code>COLORS-1</code>.</p> | | | | | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>color</i></td><td>Is the number of the color for which to provide information (0 to <code>COLORS-1</code>).</td></tr><tr><td><i>red</i></td><td>Is a pointer to the RGB value for the amount of red in <i>color</i>.</td></tr><tr><td><i>green</i></td><td>Is a pointer to the RGB value for the amount of green in <i>color</i>.</td></tr><tr><td><i>blue</i></td><td>Is a pointer to the RGB value for the amount of blue in <i>color</i>.</td></tr><tr><td><i>n</i></td><td>Is the number of a color pair.</td></tr><tr><td><i>pair</i></td><td>Is the number of the color pair for which to provide information (1 to <code>COLOR_PAIRS-1</code>).</td></tr><tr><td><i>f</i></td><td>Is a pointer to the number of the foreground color (0 to <code>COLORS-1</code>) in <i>pair</i>.</td></tr></table> | <i>color</i> | Is the number of the color for which to provide information (0 to <code>COLORS-1</code>). | <i>red</i> | Is a pointer to the RGB value for the amount of red in <i>color</i> . | <i>green</i> | Is a pointer to the RGB value for the amount of green in <i>color</i> . | <i>blue</i> | Is a pointer to the RGB value for the amount of blue in <i>color</i> . | <i>n</i> | Is the number of a color pair. | <i>pair</i> | Is the number of the color pair for which to provide information (1 to <code>COLOR_PAIRS-1</code>). | <i>f</i> | Is a pointer to the number of the foreground color (0 to <code>COLORS-1</code>) in <i>pair</i> . |
| <i>color</i> | Is the number of the color for which to provide information (0 to <code>COLORS-1</code>). | | | | | | | | | | | | | | |
| <i>red</i> | Is a pointer to the RGB value for the amount of red in <i>color</i> . | | | | | | | | | | | | | | |
| <i>green</i> | Is a pointer to the RGB value for the amount of green in <i>color</i> . | | | | | | | | | | | | | | |
| <i>blue</i> | Is a pointer to the RGB value for the amount of blue in <i>color</i> . | | | | | | | | | | | | | | |
| <i>n</i> | Is the number of a color pair. | | | | | | | | | | | | | | |
| <i>pair</i> | Is the number of the color pair for which to provide information (1 to <code>COLOR_PAIRS-1</code>). | | | | | | | | | | | | | | |
| <i>f</i> | Is a pointer to the number of the foreground color (0 to <code>COLORS-1</code>) in <i>pair</i> . | | | | | | | | | | | | | | |

COLOR_PAIR(3XCURSES)

b Is a pointer to the number of the background color (0 to COLORS-1) in *pair*.

value Is a color attribute value.

RETURN VALUES The `has_colors()` function returns `TRUE` if the terminal can manipulate colors. Otherwise, it returns `FALSE`.

The `can_change_color()` function returns `TRUE` if the terminal supports colors and is able to change their definitions. Otherwise, it returns `FALSE`.

Upon successful completion, the other functions return `OK`. Otherwise, they return `ERR`.

ERRORS No errors are defined.

USAGE To use these functions, `start_color()` must be called, usually right after `initscr(3XCURSES)`.

The `can_change_color()` and `has_colors()` functions facilitate writing terminal-independent applications. For example, a programmer can use them to decide whether to use color or some other video attribute.

On color terminals, a typical value of `COLORS` is 8 and the macros such as `COLOR_BLACK` return a value within the range from 0 to and including 7. However, applications cannot rely on this to be true.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attroff(3XCURSES)`, `delscreen(3XCURSES)`, `initscr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

COLOR_PAIRS(3XCURSES)

| | |
|------------------------------|---|
| NAME | <code>can_change_color</code> , <code>color_content</code> , <code>COLOR_PAIR</code> , <code>has_colors</code> , <code>init_color</code> , <code>init_pair</code> , <code>pair_content</code> , <code>PAIR_NUMBER</code> , <code>start_color</code> , <code>COLOR_PAIRS</code> , <code>COLORS</code> – manipulate color information |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> bool can_change_color(void); int color_content(short color, short *red, short *green, short *blue); int COLOR_PAIR(int n); bool has_colors(void); int init_color(short color, short red, short green, short blue); int init_pair(short pair, short f, short b); int pair_content(short pair, short *f, short *b); int PAIR_NUMBER(int value); int start_color(void); extern int COLOR_PAIRS; extern int COLORS;</pre> |
| DESCRIPTION | These functions manipulate color on terminals that support color. |
| Querying Capabilities | The <code>has_colors()</code> function indicates whether the terminal is a color terminal. The <code>can_change_color()</code> function indicates whether the terminal is a color terminal on which colors can be redefined. |
| Initialization | <p>The <code>start_color()</code> function must be called to enable use of colors and before any color manipulation function is called. The function initializes eight basic colors (black, red, green, yellow, blue, magenta, cyan, and white) that can be specified by the color macros (such as <code>COLOR_BLACK</code>) defined in <code><curses.h></code>. The initial appearance of these colors is unspecified.</p> <p>The function also initializes two global external variables:</p> <ul style="list-style-type: none">■ <code>COLORS</code> defines the number of colors that the terminal supports. See Color Identification below. If <code>COLORS</code> is 0, the terminal does not support redefinition of colors and <code>can_change_color()</code> will return <code>FALSE</code>.■ <code>COLOR_PAIRS</code> defines the maximum number of color-pairs that the terminal supports. See User-defined Color Pairs below. |

Color Identification

The `start_color()` function also restores the colors on the terminal to terminal-specific initial values. The initial background color is assumed to be black for all terminals.

The `init_color()` function redefines color number *color*, on terminals that support the redefinition of colors, to have the red, green, and blue intensity components specified by *red*, *green*, and *blue*, respectively. Calling `init_color()` also changes all occurrences of the specified color on the screen to the new definition.

The `color_content()` function identifies the intensity components of color number *color*. It stores the red, green, and blue intensity components of this color in the addresses pointed to by *red*, *green*, and *blue*, respectively.

For both functions, the *color* argument must be in the range from 0 to and including `COLORS-1`. Valid intensity value range from 0 (no intensity component) up to and including 1000 (maximum intensity in that component).

User-defined Color Pairs

Calling `init_pair()` defines or redefines color-pair number *pair* to have foreground color *f* and background color *b*. Calling `init_pair()` changes any characters that were displayed in the color pair's old definition to the new definition and refreshes the screen.

After defining the color pair, the macro `COLOR_PAIR(n)` returns the value of color pair *n*. This value is the color attribute as it would be extracted from a `chtype`. Controversy, the macro `COLOR_NUMBER(value)` returns the color pair number associated with the color attribute *value*.

The `pair_content()` retrieves the component colors of a color-pair number *pair*. It stores the foreground and background color numbers in the variables pointed to by *f* and *b*, respectively.

With `init_pair()` and `pair_content()`, the value of *pair* must be in a range from 0 to and including `COLOR_PAIRS-1`. Valid values for *f* and *b* are the range from 0 to and including `COLORS-1`.

PARAMETERS

| | |
|--------------|--|
| <i>color</i> | Is the number of the color for which to provide information (0 to <code>COLORS-1</code>). |
| <i>red</i> | Is a pointer to the RGB value for the amount of red in <i>color</i> . |
| <i>green</i> | Is a pointer to the RGB value for the amount of green in <i>color</i> . |
| <i>blue</i> | Is a pointer to the RGB value for the amount of blue in <i>color</i> . |
| <i>n</i> | Is the number of a color pair. |
| <i>pair</i> | Is the number of the color pair for which to provide information (1 to <code>COLOR_PAIRS-1</code>). |
| <i>f</i> | Is a pointer to the number of the foreground color (0 to <code>COLORS-1</code>) in <i>pair</i> . |

COLOR_PAIRS(3XCURSES)

b Is a pointer to the number of the background color (0 to COLORS-1) in *pair*.

value Is a color attribute value.

RETURN VALUES The `has_colors()` function returns `TRUE` if the terminal can manipulate colors. Otherwise, it returns `FALSE`.

The `can_change_color()` function returns `TRUE` if the terminal supports colors and is able to change their definitions. Otherwise, it returns `FALSE`.

Upon successful completion, the other functions return `OK`. Otherwise, they return `ERR`.

ERRORS No errors are defined.

USAGE To use these functions, `start_color()` must be called, usually right after `initscr(3XCURSES)`.

The `can_change_color()` and `has_colors()` functions facilitate writing terminal-independent applications. For example, a programmer can use them to decide whether to use color or some other video attribute.

On color terminals, a typical value of `COLORS` is 8 and the macros such as `COLOR_BLACK` return a value within the range from 0 to and including 7. However, applications cannot rely on this to be true.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attroff(3XCURSES)`, `delscreen(3XCURSES)`, `initscr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|------------------------------|--|
| NAME | can_change_color, color_content, COLOR_PAIR, has_colors, init_color, init_pair, pair_content, PAIR_NUMBER, start_color, COLOR_PAIRS, COLORS – manipulate color information |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> bool can_change_color(void); int color_content(short color, short *red, short *green, short *blue); int COLOR_PAIR(int n); bool has_colors(void); int init_color(short color, short red, short green, short blue); int init_pair(short pair, short f, short b); int pair_content(short pair, short *f, short *b); int PAIR_NUMBER(int value); int start_color(void); extern int COLOR_PAIRS; extern int COLORS;</pre> |
| DESCRIPTION | These functions manipulate color on terminals that support color. |
| Querying Capabilities | The has_colors() function indicates whether the terminal is a color terminal. The can_change_color() function indicates whether the terminal is a color terminal on which colors can be redefined. |
| Initialization | <p>The start_color() function must be called to enable use of colors and before any color manipulation function is called. The function initializes eight basic colors (black, red, green, yellow, blue, magenta, cyan, and white) that can be specified by the color macros (such as COLOR_BLACK) defined in <curses.h>. The initial appearance of these colors is unspecified.</p> <p>The function also initializes two global external variables:</p> <ul style="list-style-type: none"> ■ COLORS defines the number of colors that the terminal supports. See Color Identification below. If COLORS is 0, the terminal does not support redefinition of colors and can_change_color() will return FALSE. ■ COLOR_PAIRS defines the maximum number of color-pairs that the terminal supports. See User-defined Color Pairs below. |

COLORS(3XCURSES)

| | | | | | | | | | | | | | | | |
|---------------------------------|--|--------------|--|------------|---|--------------|---|-------------|--|----------|--------------------------------|-------------|--|----------|---|
| | <p>The <code>start_color()</code> function also restores the colors on the terminal to terminal-specific initial values. The initial background color is assumed to be black for all terminals.</p> | | | | | | | | | | | | | | |
| Color Identification | <p>The <code>init_color()</code> function redefines color number <i>color</i>, on terminals that support the redefinition of colors, to have the red, green, and blue intensity components specified by <i>red</i>, <i>green</i>, and <i>blue</i>, respectively. Calling <code>init_color()</code> also changes all occurrences of the specified color on the screen to the new definition.</p> <p>The <code>color_content()</code> function identifies the intensity components of color number <i>color</i>. It stores the red, green, and blue intensity components of this color in the addresses pointed to by <i>red</i>, <i>green</i>, and <i>blue</i>, respectively.</p> <p>For both functions, the <i>color</i> argument must be in the range from 0 to and including <code>COLORS-1</code>. Valid intensity value range from 0 (no intensity component) up to and including 1000 (maximum intensity in that component).</p> | | | | | | | | | | | | | | |
| User-defined Color Pairs | <p>Calling <code>init_pair()</code> defines or redefines color-pair number <i>pair</i> to have foreground color <i>f</i> and background color <i>b</i>. Calling <code>init_pair()</code> changes any characters that were displayed in the color pair's old definition to the new definition and refreshes the screen.</p> <p>After defining the color pair, the macro <code>COLOR_PAIR(n)</code> returns the value of color pair <i>n</i>. This value is the color attribute as it would be extracted from a <code>chtype</code>. Controversy, the macro <code>COLOR_NUMBER(value)</code> returns the color pair number associated with the color attribute <i>value</i>.</p> <p>The <code>pair_content()</code> retrieves the component colors of a color-pair number <i>pair</i>. It stores the foreground and background color numbers in the variables pointed to by <i>f</i> and <i>b</i>, respectively.</p> <p>With <code>init_pair()</code> and <code>pair_content()</code>, the value of <i>pair</i> must be in a range from 0 to and including <code>COLOR_PAIRS-1</code>. Valid values for <i>f</i> and <i>b</i> are the range from 0 to and including <code>COLORS-1</code>.</p> | | | | | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>color</i></td><td>Is the number of the color for which to provide information (0 to <code>COLORS-1</code>).</td></tr><tr><td><i>red</i></td><td>Is a pointer to the RGB value for the amount of red in <i>color</i>.</td></tr><tr><td><i>green</i></td><td>Is a pointer to the RGB value for the amount of green in <i>color</i>.</td></tr><tr><td><i>blue</i></td><td>Is a pointer to the RGB value for the amount of blue in <i>color</i>.</td></tr><tr><td><i>n</i></td><td>Is the number of a color pair.</td></tr><tr><td><i>pair</i></td><td>Is the number of the color pair for which to provide information (1 to <code>COLOR_PAIRS-1</code>).</td></tr><tr><td><i>f</i></td><td>Is a pointer to the number of the foreground color (0 to <code>COLORS-1</code>) in <i>pair</i>.</td></tr></table> | <i>color</i> | Is the number of the color for which to provide information (0 to <code>COLORS-1</code>). | <i>red</i> | Is a pointer to the RGB value for the amount of red in <i>color</i> . | <i>green</i> | Is a pointer to the RGB value for the amount of green in <i>color</i> . | <i>blue</i> | Is a pointer to the RGB value for the amount of blue in <i>color</i> . | <i>n</i> | Is the number of a color pair. | <i>pair</i> | Is the number of the color pair for which to provide information (1 to <code>COLOR_PAIRS-1</code>). | <i>f</i> | Is a pointer to the number of the foreground color (0 to <code>COLORS-1</code>) in <i>pair</i> . |
| <i>color</i> | Is the number of the color for which to provide information (0 to <code>COLORS-1</code>). | | | | | | | | | | | | | | |
| <i>red</i> | Is a pointer to the RGB value for the amount of red in <i>color</i> . | | | | | | | | | | | | | | |
| <i>green</i> | Is a pointer to the RGB value for the amount of green in <i>color</i> . | | | | | | | | | | | | | | |
| <i>blue</i> | Is a pointer to the RGB value for the amount of blue in <i>color</i> . | | | | | | | | | | | | | | |
| <i>n</i> | Is the number of a color pair. | | | | | | | | | | | | | | |
| <i>pair</i> | Is the number of the color pair for which to provide information (1 to <code>COLOR_PAIRS-1</code>). | | | | | | | | | | | | | | |
| <i>f</i> | Is a pointer to the number of the foreground color (0 to <code>COLORS-1</code>) in <i>pair</i> . | | | | | | | | | | | | | | |

b Is a pointer to the number of the background color (0 to COLORS-1) in *pair*.

value Is a color attribute value.

RETURN VALUES The `has_colors()` function returns `TRUE` if the terminal can manipulate colors. Otherwise, it returns `FALSE`.

The `can_change_color()` function returns `TRUE` if the terminal supports colors and is able to change their definitions. Otherwise, it returns `FALSE`.

Upon successful completion, the other functions return `OK`. Otherwise, they return `ERR`.

ERRORS No errors are defined.

USAGE To use these functions, `start_color()` must be called, usually right after `initscr(3XCURSES)`.

The `can_change_color()` and `has_colors()` functions facilitate writing terminal-independent applications. For example, a programmer can use them to decide whether to use color or some other video attribute.

On color terminals, a typical value of `COLORS` is 8 and the macros such as `COLOR_BLACK` return a value within the range from 0 to and including 7. However, applications cannot rely on this to be true.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attroff(3XCURSES)`, `delscreen(3XCURSES)`, `initscr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

color_set(3XCURSES)

| | | | | | | | | | |
|--------------------|--|--------------------|--|--------------------|---------------------------------------|-------------------|-----------------------------|------------------|---|
| NAME | <code>attr_get</code> , <code>attr_off</code> , <code>attr_on</code> , <code>attr_set</code> , <code>color_set</code> , <code>wattr_get</code> , <code>wattr_off</code> , <code>wattr_on</code> , <code>wattr_set</code> , <code>wcolor_set</code> – control window attributes | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int attr_get(attr_t *attrs, short *color, void *opts); int attr_off(attr_t attrs, void *opts); int attr_on(attr_t attrs, void *opts); int attr_set(attr_t attrs, short color, void *opts); int color_set(short *color, void *opts); int wattr_get(WINDOW *win, attr_t attrs, short *color, void *opts); int wattr_off(WINDOW *win, attr_t attrs, void *opts); int wattr_on(WINDOW *win, attr_t attrs, void *opts); int wattr_set(WINDOW *win, attr_t attrs, short color, void *opts); int wcolor_set(WINDOW *win, short color, void *opts);</pre> | | | | | | | | |
| DESCRIPTION | <p>The <code>attr_get()</code> function retrieves the current rendition of <code>stdscr</code>. The <code>wattr_get()</code> function retrieves the current rendition of window <code>win</code>. If <code>attrs</code> or <code>color</code> is a null pointer, no information is retrieved.</p> <p>The <code>attr_off()</code> and <code>attr_on()</code> functions unset and set, respectively, the specified window attributes of <code>stdscr</code>. These functions only affect the attributes specified; attributes that existed before the call are retained.</p> <p>The <code>wattr_off()</code> and <code>wattr_on()</code> functions unset or set the specified attributes for window <code>win</code>.</p> <p>The <code>attr_set()</code> and <code>wattr_set()</code> functions change the rendition of <code>stdscr</code> and <code>win</code>; the old values are not retained.</p> <p>The <code>color_set()</code> and <code>wcolor_set()</code> functions set the window color of <code>stdscr</code> and <code>win</code> to <code>color</code>.</p> <p>The attributes and color pairs that can be used are specified in the <code>Attributes</code>, <code>Color Pairs</code>, and <code>Renditions</code> section of the <code>curses(3XCURSES)</code> man page.</p> | | | | | | | | |
| PARAMETERS | <table><tr><td><code>attrs</code></td><td>Is a pointer to the foreground window attributes to be set or unset.</td></tr><tr><td><code>color</code></td><td>Is a pointer to a color pair number .</td></tr><tr><td><code>opts</code></td><td>Is reserved for future use.</td></tr><tr><td><code>win</code></td><td>Is a pointer to the window in which attribute changes are to be made.</td></tr></table> | <code>attrs</code> | Is a pointer to the foreground window attributes to be set or unset. | <code>color</code> | Is a pointer to a color pair number . | <code>opts</code> | Is reserved for future use. | <code>win</code> | Is a pointer to the window in which attribute changes are to be made. |
| <code>attrs</code> | Is a pointer to the foreground window attributes to be set or unset. | | | | | | | | |
| <code>color</code> | Is a pointer to a color pair number . | | | | | | | | |
| <code>opts</code> | Is reserved for future use. | | | | | | | | |
| <code>win</code> | Is a pointer to the window in which attribute changes are to be made. | | | | | | | | |

color_set(3XCURSES)

RETURN VALUES These functions always return OK.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO add_wch(3XCURSES), addnwstr(3XCURSES), attroff(3XCURSES), bkgrndset(3XCURSES), curses(3XCURSES), init_color(3XCURSES), libcurses(3XCURSES), start_color(3XCURSES), attributes(5), standards(5)

COLS(3XCURSES)

NAME COLS – number of columns on terminal screen

SYNOPSIS `cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [library...]`

```
c89 [ flag... ] file... -lcurses [ library ... ]
```

```
#include <curses.h>
```

```
extern int COLS;
```

DESCRIPTION The external variable COLS indicates the number of columns on the terminal screen.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `initscr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | plot, arc, box, circle, closepl, closevt, cont, erase, label, line, linemod, move, openpl, openvt, point, space – graphics interface |
| SYNOPSIS | <pre>cc [flag ...] file ... -lplot [library...] #include <plot.h> void arc(short x0, short y0, short x1, short y1, short x2, short y2); void box(short x0, short y0, short x1, short y1); void circle(short x, short y, short r); void closepl(); void closevt(); void cont(short x, short y); void erase(); void label(char *s); void line(short x0, short y0, short x1, short y1); void linemod(char *s); void move(short x, short y); void openpl(); void openvt(); void point(short x, short y); void space(short x0, short y0, short x1, short y1);</pre> |
| DESCRIPTION | <p>These functions generate graphics output for a set of output devices. The format of the output is dependent upon which link editor option is used when the program is compiled and linked (see Link Editor).</p> <p>The term "current point" refers to the current setting for the x and y coordinates.</p> <p>The <code>arc()</code> function specifies a circular arc. The coordinates $(x0, y0)$ specify the center of the arc. The coordinates $(x1, y1)$ specify the starting point of the arc. The coordinates $(x2, y2)$ specify the end point of the circular arc.</p> <p>The <code>box()</code> function specifies a rectangle with coordinates $(x0, y0)$, $(x0, y1)$, $(x1, y0)$, and $(x1, y1)$. The current point is set to $(x1, y1)$.</p> <p>The <code>circle()</code> function specifies a circle with a center at the coordinates (x, y) and a radius of r.</p> <p>The <code>closevt()</code> and <code>closepl()</code> functions flush the output.</p> <p>The <code>cont()</code> function specifies a line beginning at the current point and ending at the coordinates (x, y). The current point is set to (x, y).</p> |

cont(3PLOT)

The `erase()` function starts another frame of output.

The `label()` function places the null terminated string `s` so that the first character falls on the current point. The string is then terminated by a NEWLINE character.

The `line()` function draws a line starting at the coordinates (x_0, y_0) and ending at the coordinates (x_1, y_1) . The current point is set to (x_1, y_1) .

The `linemod()` function specifies the style for drawing future lines. `s` may contain one of the following: `dotted`, `solid`, `longdashed`, `shortdashed`, or `dotdashed`.

The `move()` function sets the current point to the coordinates (x, y) .

The `openpl()` or `openvt()` function must be called to open the device before any other plot functions are called.

The `point()` function plots the point given by the coordinates (x, y) . The current point is set to (x, y) .

The `space()` function specifies the size of the plotting area. The plot will be reduced or enlarged as necessary to fit the area specified. The coordinates (x_0, y_0) specify the lower left hand corner of the plotting area. The coordinates (x_1, y_1) specify the upper right hand corner of the plotting area.

Link Editor Various flavors of these functions exist for different output devices. They are obtained by using the following `ld(1)` options:

| | |
|---------------------|--|
| <code>-lplot</code> | device-independent graphics stream on standard output in the format described in <code>plot(4B)</code> |
| <code>-l300</code> | GSI 300 terminal |
| <code>-l300s</code> | GSI 300S terminal |
| <code>-l4014</code> | Tektronix 4014 terminal |
| <code>-l450</code> | GSI 450 terminal |
| <code>-lvt0</code> | |

| | |
|--------------|--|
| FILES | <code>/usr/lib/libplot.a</code> archive library |
| | <code>/usr/lib/libplot.so.1</code> shared object |
| | <code>/usr/lib/sparcv9/libplot.so.1</code> 64-bit shared object |
| | <code>/usr/lib/lib300.a</code> archive library |
| | <code>/usr/lib/lib300.so.1</code> shared object |
| | |

```

/usr/lib/sparcv9/lib300.so.1
  64-bit shared object
/usr/lib/lib300s.a
  archive library
/usr/lib/lib300s.so.1
  shared object
/usr/lib/sparcv9/lib300s.so.1
  64-bit shared object
/usr/lib/lib4014.a
  archive library
/usr/lib/lib4014.so.1
  shared object
/usr/lib/sparcv9/lib4014.so.1
  64-bit shared object
/usr/lib/lib450.a
  archive library
/usr/lib/lib450.so.1
  shared object
/usr/lib/sparcv9/lib450.so.1
  64-bit shared object
/usr/lib/libvt0.a
  archive library
/usr/lib/libvt0.so.1
  shared object
/usr/lib/sparcv9/libvt0.so.1
  64-bit shared object

```

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `graph(1)`, `ld(1)`, `libplot(3LIB)`, `plot(4B)`, `attributes(5)`

copywin(3CURSES)

NAME | curs_overlay, overlay, overwrite, copywin – overlap and manipulate overlapped curses windows

SYNOPSIS |

```
cc [ flag ... ] file ... -lcurses [ library ... ]
#include <curses.h>

int overlay(WINDOW *srcwin, WINDOW *dstwin);
int overwrite(WINDOW *srcwin, WINDOW *dstwin);

int copywin(WINDOW *srcwin, WINDOW *dstwin, int sminrow, int smincol,
            int dminrow, int dmincol, int dmaxrow, int dmaxcol, int overlay);
```

DESCRIPTION | The overlay() and overwrite() routines overlay *srcwin* on top of *dstwin*. *srcwin* and *dstwin* are not required to be the same size; only text where the two windows overlap is copied. The difference is that overlay() is non-destructive (blanks are not copied) whereas overwrite() is destructive.

The copywin() routine provides a finer granularity of control over the overlay() and overwrite() routines. Like in the refresh() routine, a rectangle is specified in the destination window, (*dminrow*, *dmincol*) and (*dmaxrow*, *dmaxcol*), and the upper-left-corner coordinates of the source window, (*sminrow*, *smincol*). If the argument *overlay* is true, then copying is non-destructive, as in overlay().

RETURN VALUES | Routines that return an integer return ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | curs_pad(3CURSES), curs_refresh(3CURSES), curses(3CURSES), attributes(5)

NOTES | The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.

Note that overlay() and overwrite may be macros.

| | | | | | | | | | | | | | | | | | | | |
|----------------------|---|---------------|---|---------------|---|----------------|---|----------------|--|----------------|---|----------------|--|----------------|--|----------------|---|----------------|---|
| NAME | copywin – overlay or overwrite any portion of window | | | | | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int copywin(const WINDOW *srcwin, WINDOW *dstwin, int sminrow, int smincol, int dminrow, int dmincol, int dmaxrow, int dmaxcol, int overlay);</pre> | | | | | | | | | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="vertical-align: top;"><i>srcwin</i></td> <td>Is a pointer to the source window to be copied.</td> </tr> <tr> <td style="vertical-align: top;"><i>dstwin</i></td> <td>Is a pointer to the destination window to be overlaid or overwritten.</td> </tr> <tr> <td style="vertical-align: top;"><i>sminrow</i></td> <td>Is the row coordinate of the upper left corner of the rectangular area on the source window to be copied.</td> </tr> <tr> <td style="vertical-align: top;"><i>smincol</i></td> <td>Is the column coordinate of the upper left corner of the rectangular area on the source window to be copied.</td> </tr> <tr> <td style="vertical-align: top;"><i>dminrow</i></td> <td>Is the row coordinate of the upper left corner of the rectangular area on the destination window to be overlaid or overwritten.</td> </tr> <tr> <td style="vertical-align: top;"><i>dmincol</i></td> <td>Is the column coordinate of the upper left corner of the rectangular area on destination window to be overlaid or overwritten.</td> </tr> <tr> <td style="vertical-align: top;"><i>dmaxrow</i></td> <td>Is the row coordinate of the lower right corner of the rectangular area on the destination window to be overlaid or overwritten.</td> </tr> <tr> <td style="vertical-align: top;"><i>dmaxcol</i></td> <td>Is the column coordinate of the lower right corner of the rectangular area on the destination window to be overlaid or overwritten.</td> </tr> <tr> <td style="vertical-align: top;"><i>overlay</i></td> <td>Is a TRUE or FALSE value that determines whether the destination window is overlaid or overwritten.</td> </tr> </table> | <i>srcwin</i> | Is a pointer to the source window to be copied. | <i>dstwin</i> | Is a pointer to the destination window to be overlaid or overwritten. | <i>sminrow</i> | Is the row coordinate of the upper left corner of the rectangular area on the source window to be copied. | <i>smincol</i> | Is the column coordinate of the upper left corner of the rectangular area on the source window to be copied. | <i>dminrow</i> | Is the row coordinate of the upper left corner of the rectangular area on the destination window to be overlaid or overwritten. | <i>dmincol</i> | Is the column coordinate of the upper left corner of the rectangular area on destination window to be overlaid or overwritten. | <i>dmaxrow</i> | Is the row coordinate of the lower right corner of the rectangular area on the destination window to be overlaid or overwritten. | <i>dmaxcol</i> | Is the column coordinate of the lower right corner of the rectangular area on the destination window to be overlaid or overwritten. | <i>overlay</i> | Is a TRUE or FALSE value that determines whether the destination window is overlaid or overwritten. |
| <i>srcwin</i> | Is a pointer to the source window to be copied. | | | | | | | | | | | | | | | | | | |
| <i>dstwin</i> | Is a pointer to the destination window to be overlaid or overwritten. | | | | | | | | | | | | | | | | | | |
| <i>sminrow</i> | Is the row coordinate of the upper left corner of the rectangular area on the source window to be copied. | | | | | | | | | | | | | | | | | | |
| <i>smincol</i> | Is the column coordinate of the upper left corner of the rectangular area on the source window to be copied. | | | | | | | | | | | | | | | | | | |
| <i>dminrow</i> | Is the row coordinate of the upper left corner of the rectangular area on the destination window to be overlaid or overwritten. | | | | | | | | | | | | | | | | | | |
| <i>dmincol</i> | Is the column coordinate of the upper left corner of the rectangular area on destination window to be overlaid or overwritten. | | | | | | | | | | | | | | | | | | |
| <i>dmaxrow</i> | Is the row coordinate of the lower right corner of the rectangular area on the destination window to be overlaid or overwritten. | | | | | | | | | | | | | | | | | | |
| <i>dmaxcol</i> | Is the column coordinate of the lower right corner of the rectangular area on the destination window to be overlaid or overwritten. | | | | | | | | | | | | | | | | | | |
| <i>overlay</i> | Is a TRUE or FALSE value that determines whether the destination window is overlaid or overwritten. | | | | | | | | | | | | | | | | | | |
| DESCRIPTION | <p>The <code>copywin()</code> function provides a finer granularity of control over the <code>overlay(3XCURSES)</code> and <code>overwrite(3XCURSES)</code> functions. As in the <code>prefresh()</code> function (see <code>newpad(3XCURSES)</code>), a rectangle is specified in the destination window, (<i>dminrow</i>, <i>dmincol</i>) and (<i>dmaxrow</i>, <i>dmaxcol</i>), and the upper-left-corner coordinates of the source window, (<i>smincol</i>, <i>sminrow</i>). If <i>overlay</i> is TRUE, then copying is non-destructive, as in <code>overlay()</code>. If <i>overlay</i> is FALSE, then copying is destructive, as in <code>overwrite()</code>.</p> | | | | | | | | | | | | | | | | | | |
| RETURN VALUES | Upon successful completion, the <code>copywin()</code> function returns OK. Otherwise, it returns ERR. | | | | | | | | | | | | | | | | | | |

copywin(3XCURSES)

ERRORS No errors are defined.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `curses(3XCURSES)`, `libcurses(3XCURSES)`, `newpad(3XCURSES)`, `overlay(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | form_page, set_form_page, set_current_field, current_field, field_index – set forms current page and field | | | | | | | | | | | | |
|----------------------|---|----------------|-------------------------------------|----------------|---------------|----------------|---------------------------|-------------|--|-----------------|---------------------------------|------------------|--------------------------------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_form_page(FORM *form, int page); int form_page(FORM *form); int set_current_field(FORM *form, FIELD *field); FIELD *current_field(FORM*form); int field_index(FIELD *field);</pre> | | | | | | | | | | | | |
| DESCRIPTION | <p>set_form_page() sets the page number of <i>form</i> to <i>page</i>. form_page() returns the current page number of <i>form</i>.</p> <p>set_current_field() sets the current field of <i>form</i> to <i>field</i>. current_field() returns a pointer to the current field of <i>form</i>.</p> <p>field_index() returns the index in the field pointer array of <i>field</i>.</p> | | | | | | | | | | | | |
| RETURN VALUES | <p>form_page() returns -1 on error.</p> <p>current_field() returns NULL on error.</p> <p>field_index() returns -1 on error.</p> <p>set_form_page() and set_current_field() return one of the following:</p> <table border="0"> <tr> <td>E_OK</td> <td>The function returned successfully.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> <tr> <td>E_BAD_ARGUMENT</td> <td>An argument is incorrect.</td> </tr> <tr> <td>E_BAD_STATE</td> <td>The routine was called from an initialization or termination function.</td> </tr> <tr> <td>E_INVALID_FIELD</td> <td>The field contents are invalid.</td> </tr> <tr> <td>E_REQUEST_DENIED</td> <td>The form driver request failed</td> </tr> </table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An argument is incorrect. | E_BAD_STATE | The routine was called from an initialization or termination function. | E_INVALID_FIELD | The field contents are invalid. | E_REQUEST_DENIED | The form driver request failed |
| E_OK | The function returned successfully. | | | | | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | | | | | |
| E_BAD_ARGUMENT | An argument is incorrect. | | | | | | | | | | | | |
| E_BAD_STATE | The routine was called from an initialization or termination function. | | | | | | | | | | | | |
| E_INVALID_FIELD | The field contents are invalid. | | | | | | | | | | | | |
| E_REQUEST_DENIED | The form driver request failed | | | | | | | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | | | | | | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | | | | | | | |
| MT-Level | Unsafe | | | | | | | | | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | | | | | | | | | |

current_field(3CURSES)

NOTES | The header `<form.h>` automatically includes the headers `<eti.h>` and `<curses.h>`.

current_item(3CURSES)

NAME menu_item_current, set_current_item, current_item, set_top_row, top_row, item_index
– set and get current menus items

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

int set_current_item(MENU *menu, ITEM *item);
ITEM *current_item(MENU *menu);
int set_top_row(MENU *menu, int row);
int top_row(MENU *menu);
int item_index(ITEM *item);
```

DESCRIPTION The current item of a menu is the item where the cursor is currently positioned. set_current_item() sets the current item of *menu* to *item*. current_item() returns a pointer to the the current item in *menu*.

set_top_row() sets the top row of *menu* to *row*. The left-most item on the new top row becomes the current item. top_row() returns the number of the menu row currently displayed at the top of *menu*.

item_index() returns the index to the *item* in the item pointer array. The value of this index ranges from 0 through *N*-1, where *N* is the total number of items connected to the menu.

RETURN VALUES current_item() returns NULL on error.

top_row() and index_item() return -1 on error.

set_current_item() and set_top_row() return one of the following:

| | |
|-----------------|--|
| E_OK | The routine returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. |
| E_BAD_STATE | The routine was called from an initialization or termination function. |
| E_NOT_CONNECTED | No items are connected to the menu. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), attributes(5)

current_item(3CURSES)

NOTES | The header `<menu.h>` automatically includes the headers `<eti.h>` and `<curses.h>`.

| | |
|----------------------|--|
| NAME | curs_addch, addch, waddch, mvaddch, mvwaddch, echochar, wechochar – add a character (with attributes) to a curses window and advance cursor |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int addch(chtype ch); int waddch(WINDOW *win, chtype ch); int mvaddch(int y, int x, chtype ch); int mvwaddch(WINDOW *win, int y, int x, chtype ch); int echochar(chtype ch); int wechochar(WINDOW *win, chtype ch);</pre> |
| DESCRIPTION | <p>With the <code>addch()</code>, <code>waddch()</code>, <code>mvaddch()</code>, and <code>mvwaddch()</code> routines, the character <code>ch</code> is put into the window at the current cursor position of the window and the position of the window cursor is advanced. Its function is similar to that of <code>putchar()</code>. At the right margin, an automatic newline is performed. At the bottom of the scrolling region, if <code>scrollok()</code> is enabled, the scrolling region is scrolled up one line.</p> <p>If <code>ch</code> is a tab, newline, or backspace, the cursor is moved appropriately within the window. A newline also does a <code>clrtoeol()</code> before moving. Tabs are considered to be at every eighth column. If <code>ch</code> is another control character, it is drawn in the <code>^X</code> notation. Calling <code>winch()</code> after adding a control character does not return the control character, but instead returns the representation of the control character. See <code>curs_inch(3CURSES)</code>.</p> <p>Video attributes can be combined with a character by OR-ing them into the parameter. This results in these attributes also being set. (The intent here is that text, including attributes, can be copied from one place to another using <code>inch()</code> and <code>addch()</code>.) (see <code>standout()</code>, predefined video attribute constants, on the <code>curs_attr(3CURSES)</code> page).</p> <p>The <code>echochar()</code> and <code>wechochar()</code> routines are functionally equivalent to a call to <code>addch()</code> followed by a call to <code>refresh()</code>, or a call to <code>waddch</code> followed by a call to <code>wrefresh()</code>. The knowledge that only a single character is being output is taken into consideration and, for non-control characters, a considerable performance gain might be seen by using these routines instead of their equivalents.</p> |
| Line Graphics | <p>The following variables may be used to add line drawing characters to the screen with routines of the <code>addch()</code> family. When variables are defined for the terminal, the <code>A_ALTCHARSET</code> bit is turned on (see <code>curs_attr(3CURSES)</code>). Otherwise, the default character listed below is stored in the variable. The names chosen are consistent with the VT100 nomenclature.</p> |

curs_addch(3CURSES)

| Name | Default | Glyph Description |
|--------------|---------|-------------------------|
| ACS_ULCORNER | + | upper left-hand corner |
| ACS_LLCORNER | + | lower left-hand corner |
| ACS_URCORNER | + | upper right-hand corner |
| ACS_LRCORNER | + | lower right-hand corner |
| ACS_RTEE | + | right tee |
| ACS_LTEE | + | left tee |
| ACS_BTEE | + | bottom tee |
| ACS_TTEE | + | top tee |
| ACS_HLINE | - | horizontal line |
| ACS_VLINE | | vertical line |
| ACS_PLUS | + | plus |
| ACS_S1 | - | scan line 1 |
| ACS_S9 | - | scan line 9 |
| ACS_DIAMOND | + | diamond |
| ACS_CKBOARD | : | checker board (stipple) |
| ACS_DEGREE | ' | degree symbol |
| ACS_PLMINUS | # | plus/minus |
| ACS_BULLET | o | bullet |
| ACS_LARROW | < | arrow pointing left |
| ACS_RARROW | > | arrow pointing right |
| ACS_DARROW | v | arrow pointing down |
| ACS_UARROW | ^ | arrow pointing up |
| ACS_BOARD | # | board of squares |
| ACS_LANTERN | # | lantern symbol |
| ACS_BLOCK | # | solid square block |

RETURN VALUES All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

curs_addch(3CURSES)

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_attr(3CURSES), curs_clear(3CURSES), curs_inch(3CURSES), curs_outopts(3CURSES), curs_refresh(3CURSES), curses(3CURSES), putc(3C), attributes(5)

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `addch()`, `mvaddch()`, `mvwaddch()`, and `echochar()` may be macros.

curs_addchstr(3CURSES)

| NAME | curs_addchstr, addchstr, addchnstr, waddchstr, waddchnstr, mvaddchstr, mvaddchnstr, mvwaddchstr, mvwaddchnstr – add string of characters and attributes to a curses window | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int addchstr(chtype *chstr); int addchnstr(chtype *chstr, int n); int waddchstr(WINDOW *win, chtype *chstr); int waddchnstr(WINDOW *win, chtype *chstr, int n); int mvaddchstr(int y, int x, chtype *chstr); int mvaddchnstr(int y, int x, chtype *chstr, int n); int mvwaddchstr(WINDOW *win, int y, int x, chtype *chstr); int mvwaddchnstr(WINDOW *win, int y, int x, chtype *chstr, int n);</pre> | | | | |
| DESCRIPTION | <p>All of these routines copy <i>chstr</i> directly into the window image structure starting at the current cursor position. The four routines with <i>n</i> as the last argument copy at most <i>n</i> elements, but no more than will fit on the line. If <i>n</i>=-1 then the whole string is copied, to the maximum number that fit on the line.</p> <p>The position of the window cursor is not advanced. These routines works faster than <code>waddnstr()</code> (see <code>curs_addstr(3CURSES)</code>) because they merely copy <i>chstr</i> into the window image structure. On the other hand, care must be taken when using these functions because they do not perform any kind of checking (such as for the newline character), they do not advance the current cursor position, and they truncate the string, rather than wrapping it around to the next line.</p> | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">MT-Level</td> <td style="text-align: center;">Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_addstr(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that all routines except <code>waddchnstr()</code> and <code>waddchstr()</code> may be macros.</p> | | | | |

NAME curs_addstr, addstr, addnstr, waddstr, waddnstr, mvaddstr, mvaddnstr, mvwaddstr, mvwaddnstr – add a string of characters to a curses window and advance cursor

SYNOPSIS

```
cc [ flag ... ] file ... -lcurses [ library ... ]
#include <curses.h>

int addstr(char *str);
int addnstr(char *str, int n);
int waddstr(WINDOW *win, char *str);
int waddnstr(WINDOW *win, char *str, int n);
int mvaddstr(int y, int x, char *str);
int mvaddnstr(int y, int x, char *str, int n);
int mvwaddstr(WINDOW *win, int y, int x, char *str);
int mvwaddnstr(WINDOW *win, int y, int x, char *str, int n);
```

DESCRIPTION All of these routines write all the characters of the null terminated character string *str* on the given window. It is similar to calling `waddch()` once for each character in the string. The four routines with *n* as the last argument write at most *n* characters. If *n* is negative, then the entire string will be added.

RETURN VALUES All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_addch(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that all routines except `waddstr()` and `waddnstr()` may not be macros.

curs_addwch(3CURSES)

| | |
|----------------------|--|
| NAME | <code>curs_addwch</code> , <code>addwch</code> , <code>waddwch</code> , <code>mvaddwch</code> , <code>mvwaddwch</code> , <code>echowchar</code> , <code>wechowchar</code> – add a <code>wchar_t</code> character (with attributes) to a curses window and advance cursor |
| SYNOPSIS | <pre>cc [flag...] file... -lcurses [library...] #include<curses.h> int addwch(chtype <i>wch</i>); int waddwch(WINDOW *<i>win</i>, chtype <i>wch</i>); int mvaddwch(int <i>y</i>, int <i>x</i>, chtype <i>wch</i>); int mvwaddwch(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>, chtype <i>wch</i>); int echowchar(chtype <i>wch</i>); int wechowchar(WINDOW *<i>win</i>, chtype <i>wch</i>);</pre> |
| DESCRIPTION | <p>The <code>addwch()</code>, <code>waddwch()</code>, <code>mvaddwch()</code>, and <code>mvwaddwch()</code> routines put the character <code>wch</code>, holding a <code>wchar_t</code> character, into the window at the current cursor position of the window and advance the position of the window cursor. Their function is similar to that of <code>putwchar(3C)</code> in the C multibyte library. At the right margin, an automatic newline is performed. At the bottom of the scrolling region, if <code>scrollok</code> is enabled, the scrolling region is scrolled up one line.</p> <p>If <code>wch</code> is a tab, newline, or backspace, the cursor is moved appropriately within the window. A newline also does a <code>clrtoeol(3CURSES)</code> before moving. Tabs are considered to be at every eighth column. If <code>wch</code> is another control character, it is drawn in the <code>^X</code> notation. Calling <code>winwch(3CURSES)</code> after adding a control character does not return the control character, but instead returns the representation of the control character.</p> <p>Video attributes can be combined with a <code>wchar_t</code> character by OR-ing them into the parameter. This results in these attributes also being set. (The intent here is that text, including attributes, can be copied from one place to another using <code>inwch()</code> and <code>addwch()</code>.) See <code>standout(3CURSES)</code>, predefined video attribute constants.</p> <p>The <code>echowchar()</code> and <code>wechowchar()</code> routines are functionally equivalent to a call to <code>addwch()</code> followed by a call to <code>refresh(3CURSES)</code>, or a call to <code>waddwch()</code> followed by a call to <code>wrefresh(3CURSES)</code>. The knowledge that only a single character is being output is taken into consideration and, for non-control characters, a considerable performance gain might be seen by using these routines instead of their equivalents.</p> |
| Line Graphics | <p>The following variables may be used to add line drawing characters to the screen with routines of the <code>addwch()</code> family. When variables are defined for the terminal, the <code>A_ALTCHARSET</code> bit is turned on. (See <code> curs_attr(3CURSES)</code>). Otherwise, the default character listed below is stored in the variable. The names chosen are consistent with the VT100 nomenclature.</p> |

curs_addwch(3CURSES)

| Name | Default | Glyph Description |
|--------------|---------|-------------------------|
| ACS_ULCORNER | + | upper left-hand corner |
| ACS_LLCORNER | + | lower left-hand corner |
| ACS_URCORNER | + | upper right-hand corner |
| ACS_LRCORNER | + | lower right-hand corner |
| ACS_RTEE | + | right tee |
| ACS_LTEE | + | left tee |
| ACS_BTEE | + | bottom tee |
| ACS_TTEE | + | top tee |
| ACS_HLINE | - | horizontal line |
| ACS_VLINE | | vertical line |
| ACS_PLUS | + | plus |
| ACS_S1 | - | scan line 1 |
| ACS_S9 | - | scan line 9 |
| ACS_DIAMOND | + | diamond |
| ACS_CKBOARD | : | checker board (stipple) |
| ACS_DEGREE | ' | degree symbol |
| ACS_PLMINUS | # | plus/minus |
| ACS_BULLET | o | bullet |
| ACS_LARROW | < | arrow pointing left |
| ACS_RARRROW | > | arrow pointing right |
| ACS_DARROW | v | arrow pointing down |
| ACS_UARROW | ^ | arrow pointing up |
| ACS_BOARD | # | board of squares |
| ACS_LANTERN | # | lantern symbol |
| ACS_BLOCK | # | solid square block |

RETURN VALUE

All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion, unless otherwise noted in the preceding routine descriptions.

`curs_addwch(3CURSES)`

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `putwchar(3C)`, `clrtoeol(3CURSES)`, `curses(3CURSES)`, `curs_attr(3CURSES)`, `curs_inwch(3CURSES)`, `curs_outopts(3CURSES)`, `refresh(3CURSES)`, `standout(3CURSES)`, `winwch(3CURSES)`, `wrefresh(3CURSES)`, `attributes(5)`

NOTES The header file `<curses.h>` automatically includes the header files `<stdio.h>`, `<unctrl.h>` and `<widec.h>`.

Note that `addwch()`, `mvaddwch()`, `mvwaddwch()`, and `echowchar()` may be macros.

None of these routines can use the color attribute in `chtype`.

| NAME | curs_addwchstr, addwchstr, addwchnstr, waddwchstr, waddwchnstr, mvaddwchstr, mvaddwchnstr, mvwaddwchstr, mvwaddwchnstr – add string of wchar_t characters (and attributes) to a curses window | | | | |
|---------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lcurses [library...] #include<curses.h> int addwchstr(chtype *wchstr); int addwchnstr(chtype *wchstr, int n); int waddwchstr(WINDOW *win, chtype *wchstr); int waddwchnstr(WINDOW *win, chtype *wchstr, int n); int mvaddwchstr(int y, int x, chtype *wchstr); int mvaddwchnstr(int y, int x, chtype *wchstr, int n); int mvwaddwchstr(WINDOW *win, int y, int x, chtype *wchstr); int mvwaddwchnstr(WINDOW *win, int y, int x, chtype *wchstr, int n);</pre> | | | | |
| DESCRIPTION | <p>All of these routines copy <i>wchstr</i>, which points to a string of wchar_t characters, directly into the window image structure starting at the current cursor position. The four routines with <i>n</i> as the last argument copy at most <i>n</i> elements, but no more than will fit on the line. If <i>n</i>==-1 then the whole string is copied, to the maximum number that fit on the line.</p> <p>The position of the window cursor is not advanced. These routines work faster than waddnwstr(3CURSES) because they merely copy <i>wchstr</i> into the window image structure. On the other hand, care must be taken when using these functions because they don't perform any kind of checking (such as for the newline character), they do not advance the current cursor position, and they truncate the string, rather than wrapping it around to the new line.</p> | | | | |
| RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion, unless otherwise noted in the preceding routine descriptions. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), waddnwstr(3CURSES), attributes(5) | | | | |
| NOTES | The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <widec.h>. | | | | |

`curs_addwchstr(3CURSES)`

Note that all routines except `waddwchnstr()` may be macros.

None of these routines can use the color attribute in `chtype`.

| NAME | curs_addwstr, addwstr, addnwstr, waddwstr, waddnwstr, mvaddwstr, mvaddnwstr, mvwaddwstr, mvwaddnwstr – add a string of wchar_t characters to a curses window and advance cursor | | | | |
|---------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lcurses [library...] #include<curses.h> int addwstr(wchar_t *wstr); int addnwstr(wchar_t *wstr, int n); int waddwstr(WINDOW *win, wchar_t *wstr); int waddnwstr(WINDOW *win, wchar_t *wstr, int n); int mvaddwstr(int y, int x, wchar_t *wstr); int mvaddnwstr(int y, int x, wchar_t *wstr, int n); int mvwaddwstr(WINDOW *win, int y, int x, wchar_t *wstr); int mvwaddnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);</pre> | | | | |
| DESCRIPTION | All of these routines write all the characters of the null-terminated wchar_t character string wstr on the given window. The effect is similar to calling waddwch(3CURSES) once for each wchar_t character in the string. The four routines with n as the last argument write at most n wchar_t characters. If n is negative, then the entire string will be added. | | | | |
| RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), waddwch(3CURSES), attributes(5) | | | | |
| NOTES | <p>The header file <curses.h> automatically includes the header files <stdio.h>, <nctrl.h> and <widec.h>.</p> <p>Note that all of these routines except waddwstr() and waddnwstr() may be macros.</p> | | | | |

curs_alecompat(3CURSES)

| NAME | curs_alecompat, movenextch, wmovenextch, moveprevch, wmoveprevch, adjcurspos, wadjcurspos – these functions are added to ALE curses library for moving the cursor by character. | | | | |
|---------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int movenextch(void); int wmovenextch(WINDOW *win); int moveprevch(void); int wmoveprevch(WINDOW *win); int adjcurspos(void); int wadjcurspos(WINDOW *win);</pre> | | | | |
| DESCRIPTION | <p>movenextch() and wmovenextch() move the cursor to the next character to the right. If the next character is a multicolumn character, the cursor is positioned on the first (left-most) column of that character. The new cursor position will be on the next character, even if the cursor was originally positioned on the left-most column of a multicolumn character. Note that the simple cursor increment (++x) does not guarantee movement to the next character, if the cursor was originally positioned on a multicolumn character. getyx(3CURSES) can be used to find the new position.</p> <p>moveprevch() and wmoveprevch() routines are the opposite of movenextch() and wmovenextch(), moving the cursor to the left-most column of the previous character.</p> <p>adjcurspos() and wadjcurspos() move the cursor to the first(left-most) column of the multicolumn character that the cursor is presently on. If the cursor is already on the first column, or if the cursor is on a single-column character, these routines will have no effect.</p> | | | | |
| RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), getyx(3CURSES), attributes(5) | | | | |
| NOTES | <p>The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <widec.h>.</p> <p>Note that movenextch(), moveprevch(), and adjcurspos() may be macros.</p> | | | | |

| | | | | | | | | | | | | | | | |
|--------------------|--|------------|--|-------------|-------------|-----------|---------------|---------|----------|-------|-------------|--------|----------------------|--------------|-------------------------|
| NAME | curs_attr, attroff, wattroff, attron, wattron, attrset, wattrset, standend, wstandend, standout, wstandout – curses character and window attribute control routines | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int attroff(int attrs); int wattroff(WINDOW *win, int attrs); int attron(int attrs); int wattron(WINDOW *win, int attrs); int attrset(int attrs); int wattrset(WINDOW *win, int attrs); int standend(void); int wstandend(WINDOW *win); int standout(void); int wstandout(WINDOW *win);</pre> | | | | | | | | | | | | | | |
| DESCRIPTION | <p>All of these routines manipulate the current attributes of the named window. The current attributes of a window are applied to all characters that are written into the window with <code>waddch()</code>, <code>waddstr()</code>, and <code>wprintw()</code>. Attributes are a property of the character, and move with the character through any scrolling and insert/delete line/character operations. To the extent possible on the particular terminal, they are displayed as the graphic rendition of characters put on the screen.</p> <p>The routine <code>attrset()</code> sets the current attributes of the given window to <code>attrs</code>. The routine <code>attroff()</code> turns off the named attributes without turning any other attributes on or off. The routine <code>attron()</code> turns on the named attributes without affecting any others. The routine <code>standout()</code> is the same as <code>attron(A_STANDOUT)</code>. The routine <code>standend()</code> is the same as <code>attrset()</code>, that is, it turns off all attributes.</p> | | | | | | | | | | | | | | |
| Attributes | <p>The following video attributes, defined in <code><curses.h></code>, can be passed to the routines <code>attron()</code>, <code>attroff()</code>, and <code>attrset()</code>, or OR-ed with the characters passed to <code>addch()</code>.</p> <table border="0"> <tr> <td style="padding-right: 20px;">A_STANDOUT</td> <td>Best highlighting mode of the terminal</td> </tr> <tr> <td>A_UNDERLINE</td> <td>Underlining</td> </tr> <tr> <td>A_REVERSE</td> <td>Reverse video</td> </tr> <tr> <td>A_BLINK</td> <td>Blinking</td> </tr> <tr> <td>A_DIM</td> <td>Half bright</td> </tr> <tr> <td>A_BOLD</td> <td>Extra bright or bold</td> </tr> <tr> <td>A_ALTCHARSET</td> <td>Alternate character set</td> </tr> </table> | A_STANDOUT | Best highlighting mode of the terminal | A_UNDERLINE | Underlining | A_REVERSE | Reverse video | A_BLINK | Blinking | A_DIM | Half bright | A_BOLD | Extra bright or bold | A_ALTCHARSET | Alternate character set |
| A_STANDOUT | Best highlighting mode of the terminal | | | | | | | | | | | | | | |
| A_UNDERLINE | Underlining | | | | | | | | | | | | | | |
| A_REVERSE | Reverse video | | | | | | | | | | | | | | |
| A_BLINK | Blinking | | | | | | | | | | | | | | |
| A_DIM | Half bright | | | | | | | | | | | | | | |
| A_BOLD | Extra bright or bold | | | | | | | | | | | | | | |
| A_ALTCHARSET | Alternate character set | | | | | | | | | | | | | | |

curs_attr(3CURSES)

A_CHARTEXT Bit-mask to extract a character

COLOR_PAIR(*n*) Color-pair number *n*

The following macro is the reverse of COLOR_PAIR(*n*) :

PAIR_NUMBER(*attrs*) Returns the pair number associated with the
 COLOR_PAIR(*n*) attribute

RETURN VALUES These routines always return 1.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_addch(3CURSES), curs_addstr(3CURSES), curs_printw(3CURSES),
 curses(3CURSES), attributes(5)

NOTES The header <curses.h> automatically includes the headers <stdio.h> and
 <unctrl.h>.

Note that attroff(), wattroff(), attron(), wattron(), wattrset(),
 standend(), and standout() may be macros.

NAME | curs_beep, beep, flash – curses bell and screen flash routines

SYNOPSIS | `cc [flag ...] file ... -lcurses [library ...]`
`#include <curses.h>`
`int beep(void);`
`int flash(void);`

DESCRIPTION | The `beep()` and `flash()` routines are used to signal the terminal user. The routine `beep()` sounds the audible alarm on the terminal, if possible; if that is not possible, it flashes the screen (visible bell), if that is possible. The routine `flash()` flashes the screen, and if that is not possible, sounds the audible signal. If neither signal is possible, nothing happens. Nearly all terminals have an audible signal (bell or beep), but only some can flash the screen.

RETURN VALUES | These routines always return OK.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | `curses(3CURSES)`, `attributes(5)`

NOTES | The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

curs_bkgd(3CURSES)

| NAME | curs_bkgd, bkgd, bkgdset, wbkgdset, wbkgd – curses window background manipulation routines | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int bkgd(chtype ch); void bkgdset(chtype ch); void wbkgdset(WINDOW *win, chtype ch); int wbkgd(WINDOW *win, chtype ch);</pre> | | | | |
| DESCRIPTION | <p>The <code>bkgdset()</code> and <code>wbkgdset()</code> routines manipulate the background of the named window. Background is a <code>chtype</code> consisting of any combination of attributes and a character. The attribute part of the background is combined (ORed) with all non-blank characters that are written into the window with <code>waddch()</code>. Both the character and attribute parts of the background are combined with the blank characters. The background becomes a property of the character and moves with the character through any scrolling and insert/delete line/character operations. To the extent possible on a particular terminal, the attribute part of the background is displayed as the graphic rendition of the character put on the screen.</p> <p>The <code>bkgd()</code> and <code>wbkgd()</code> routines combine the new background with every position in the window. Background is any combination of attributes and a character. Only the attribute part is used to set the background of non-blank characters, while both character and attributes are used for blank positions. To the extent possible on a particular terminal, the attribute part of the background is displayed as the graphic rendition of the character put on the screen.</p> | | | | |
| RETURN VALUES | <code>bkgd()</code> and <code>wbkgd()</code> return the integer <code>OK</code> , or a non-negative integer, if <code>immedok()</code> is set. See <code>curs_ouptos(3CURSES)</code> . | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_addch(3CURSES)</code> , <code>curs_ouptos(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that <code>bkgdset()</code> and <code>bkgd()</code> may be macros.</p> | | | | |

| | | | | | | | | | | | | | | | | | |
|--------------------|---|-----------------|-------------------------|-----------------|--------------------------|-----------------|------------------------|-----------------|---------------------------|-----------------|----------------------|-----------------|-----------------------|-----------------|-------------------------|-----------------|--------------------------|
| NAME | curs_border, border, wborder, box, whline, wvline – create curses borders, horizontal and vertical lines | | | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int border(chtype ls, chtype rs, chtype ts, chtype bs, chtype tl, chtype tr, chtype bl, chtype br); int wborder(WINDOW *win, chtype ls, chtype rs, chtype ts, chtype bs, chtype tl, chtype tr, chtype bl, chtype br); int box(WINDOW *win, chtype verch, chtype horch); int hline(chtype ch, int n); int whline(WINDOW *win, chtype ch, int n); int vline(chtype ch, int n); int wvline(WINDOW *win, chtype ch, int n);</pre> | | | | | | | | | | | | | | | | |
| DESCRIPTION | <p>With the <code>border()</code>, <code>wborder()</code>, and <code>box()</code> routines, a border is drawn around the edges of the window. The arguments and attributes are:</p> <hr/> <table border="0"> <tr> <td style="padding-right: 20px;"><code>ls</code></td> <td>left side of the border</td> </tr> <tr> <td><code>rs</code></td> <td>right side of the border</td> </tr> <tr> <td><code>ts</code></td> <td>top side of the border</td> </tr> <tr> <td><code>bs</code></td> <td>bottom side of the border</td> </tr> <tr> <td><code>tl</code></td> <td>top left-hand corner</td> </tr> <tr> <td><code>tr</code></td> <td>top right-hand corner</td> </tr> <tr> <td><code>bl</code></td> <td>bottom left-hand corner</td> </tr> <tr> <td><code>br</code></td> <td>bottom right-hand corner</td> </tr> </table> <hr/> <p>If any of these arguments is zero, then the following default values (defined in <code><curses.h></code>) are used respectively instead: <code>ACS_VLINE</code>, <code>ACS_VLINE</code>, <code>ACS_HLINE</code>, <code>ACS_HLINE</code>, <code>ACS_ULCORNER</code>, <code>ACS_URCORNER</code>, <code>ACS_BLCORNER</code>, <code>ACS_BRCORNER</code>.</p> <p><code>box(win, verch, horch)</code> is a shorthand for the following call:</p> <pre>wborder(win, verch, verch, horch, horch, 0, 0, 0, 0)</pre> <p><code>hline()</code> and <code>whline()</code> draw a horizontal (left to right) line using <code>ch</code> starting at the current cursor position in the window. The current cursor position is not changed. The line is at most <code>n</code> characters long, or as many as fit into the window.</p> | <code>ls</code> | left side of the border | <code>rs</code> | right side of the border | <code>ts</code> | top side of the border | <code>bs</code> | bottom side of the border | <code>tl</code> | top left-hand corner | <code>tr</code> | top right-hand corner | <code>bl</code> | bottom left-hand corner | <code>br</code> | bottom right-hand corner |
| <code>ls</code> | left side of the border | | | | | | | | | | | | | | | | |
| <code>rs</code> | right side of the border | | | | | | | | | | | | | | | | |
| <code>ts</code> | top side of the border | | | | | | | | | | | | | | | | |
| <code>bs</code> | bottom side of the border | | | | | | | | | | | | | | | | |
| <code>tl</code> | top left-hand corner | | | | | | | | | | | | | | | | |
| <code>tr</code> | top right-hand corner | | | | | | | | | | | | | | | | |
| <code>bl</code> | bottom left-hand corner | | | | | | | | | | | | | | | | |
| <code>br</code> | bottom right-hand corner | | | | | | | | | | | | | | | | |

`curs_border(3CURSES)`

`vline()` and `wvline()` draw a vertical (top to bottom) line using *ch* starting at the current cursor position in the window. The current cursor position is not changed. The line is at most *n* characters long, or as many as fit into the window.

RETURN VALUES All routines return the integer `OK`, or a non-negative integer if `immedok()` is set. See `curs_outopts(3CURSES)`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_outopts(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `border()` and `box()` may be macros.

NAME | curs_clear, erase, werase, clear, wclear, clrrobot, wclrrobot, clrtoeol, wclrtoeol – clear all or part of a curses window

SYNOPSIS | `cc [flag...] file ... -lcurses [library ...]
#include <curses.h>

int erase(void);
int werase(WINDOW *win);

int clear(void);
int wclear(WINDOW *win);

int clrrobot(void);
int wclrrobot(WINDOW *win);

int clrtoeol(void);
int wclrtoeol(WINDOW *win);`

DESCRIPTION | The erase() and werase() routines copy blanks to every position in the window.

The clear() and wclear() routines are like erase() and werase(), but they also call clearok(), so that the screen is cleared completely on the next call to wrefresh() for that window and repainted from scratch.

The clrrobot() and wclrrobot() routines erase all lines below the cursor in the window. Also, the current line to the right of the cursor, inclusive, is erased.

The clrtoeol() and wclrtoeol() routines erase the current line to the right of the cursor, inclusive.

RETURN VALUES | All routines return the integer OK, or a non-negative integer if immedok() is set. See curs_ouptos(3CURSES).

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | curs_ouptos(3CURSES), curs_refresh(3CURSES), curses(3CURSES), attributes(5)

NOTES | The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.

Note that erase(), werase(), clear(), wclear(), clrrobot(), and clrtoeol() may be macros.

curs_color(3CURSES)

| | |
|-----------------------------|---|
| NAME | <code>curs_color</code> , <code>start_color</code> , <code>init_pair</code> , <code>init_color</code> , <code>has_colors</code> , <code>can_change_color</code> , <code>color_content</code> , <code>pair_content</code> – curses color manipulation routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int start_color(void); int init_pair(short <i>pair</i>, short <i>fg</i>, short <i>bg</i>); int init_color(short <i>color</i>, short <i>red</i>, short <i>green</i>, short <i>blue</i>); bool has_colors(void); bool can_change_color(void); int color_content(short <i>color</i>, short *<i>redp</i>, short *<i>greenp</i>, short *<i>bluep</i>); int pair_content(short <i>pair</i>, short *<i>fgp</i>, short *<i>bgp</i>);</pre> |
| Overview | <p>curses provides routines that manipulate color on color alphanumeric terminals. To use these routines <code>start_color()</code> must be called, usually right after <code>initscr()</code>. See <code>curs_initscr(3CURSES)</code>. Colors are always used in pairs (referred to as color-pairs). A color-pair consists of a foreground color (for characters) and a background color (for the field on which the characters are displayed). A programmer initializes a color-pair with the routine <code>init_pair</code>. After it has been initialized, <code>COLOR_PAIR(<i>n</i>)</code>, a macro defined in <code><curses.h></code>, can be used in the same ways other video attributes can be used. If a terminal is capable of redefining colors, the programmer can use the routine <code>init_color()</code> to change the definition of a color. The routines <code>has_colors()</code> and <code>can_change_color()</code> return <code>TRUE</code> or <code>FALSE</code>, depending on whether the terminal has color capabilities and whether the programmer can change the colors. The routine <code>color_content()</code> allows a programmer to identify the amounts of red, green, and blue components in an initialized color. The routine <code>pair_content()</code> allows a programmer to find out how a given color-pair is currently defined.</p> |
| Routine Descriptions | <p>The <code>start_color()</code> routine requires no arguments. It must be called if the programmer wants to use colors, and before any other color manipulation routine is called. It is good practice to call this routine right after <code>initscr()</code>. <code>start_color()</code> initializes eight basic colors (black, red, green, yellow, blue, magenta, cyan, and white), and two global variables, <code>COLORS</code> and <code>COLOR_PAIRS</code> (respectively defining the maximum number of colors and color-pairs the terminal can support). It also restores the colors on the terminal to the values they had when the terminal was just turned on.</p> <p>The <code>init_pair()</code> routine changes the definition of a color-pair. It takes three arguments: the number of the color-pair to be changed, the foreground color number, and the background color number. The value of the first argument must be between 1 and <code>COLOR_PAIRS-1</code>. The value of the second and third arguments must be between 0 and <code>COLORS</code>. If the color-pair was previously initialized, the screen is refreshed and all occurrences of that color-pair is changed to the new definition.</p> |

`curs_color(3CURSES)`

The `init_color()` routine changes the definition of a color. It takes four arguments: the number of the color to be changed followed by three RGB values (for the amounts of red, green, and blue components). The value of the first argument must be between 0 and `COLORS`. (See the section `COLORS` for the default color index.) Each of the last three arguments must be a value between 0 and 1000. When `init_color()` is used, all occurrences of that color on the screen immediately change to the new definition.

The `has_colors()` routine requires no arguments. It returns `TRUE` if the terminal can manipulate colors; otherwise, it returns `FALSE`. This routine facilitates writing terminal-independent programs. For example, a programmer can use it to decide whether to use color or some other video attribute.

The `can_change_color()` routine requires no arguments. It returns `TRUE` if the terminal supports colors and can change their definitions; other, it returns `FALSE`. This routine facilitates writing terminal-independent programs.

The `color_content()` routine gives users a way to find the intensity of the red, green, and blue (RGB) components in a color. It requires four arguments: the color number, and three addresses of `shorts` for storing the information about the amounts of red, green, and blue components in the given color. The value of the first argument must be between 0 and `COLORS`. The values that are stored at the addresses pointed to by the last three arguments are between 0 (no component) and 1000 (maximum amount of component).

The `pair_content()` routine allows users to find out what colors a given color-pair consists of. It requires three arguments: the color-pair number, and two addresses of `shorts` for storing the foreground and the background color numbers. The value of the first argument must be between 1 and `COLOR_PAIRS-1`. The values that are stored at the addresses pointed to by the second and third arguments are between 0 and `COLORS`.

Colors In `<curses.h>` the following macros are defined. These are the default colors. `curses` also assumes that `COLOR_BLACK` is the default background color for all terminals.

```
COLOR_BLACK
COLOR_RED
COLOR_GREEN
COLOR_YELLOW
COLOR_BLUE
COLOR_MAGENTA
COLOR_CYAN
COLOR_WHITE
```

RETURN VALUES All routines that return an integer return `ERR` upon failure and `OK` upon successful completion.

curl_color(3CURSES)

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curl_attr(3CURSES), curl_initscr(3CURSES), curses(3CURSES), attributes(5)

NOTES The header <curls.h> automatically includes the headers <stdio.h> and <unctrl.h>.

NAME curscr – current window

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

extern WINDOW *curscr;
```

DESCRIPTION The external variable `curscr` points to an internal data structure. It can be specified as an argument to certain functions such as `clearok(3XCURSES)`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `clearok(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

curs_delch(3CURSES)

NAME curs_delch, delch, wdelch, mvdelch, mvwdelch – delete character under cursor in a curses window

SYNOPSIS

```
cc [ flag ... ] file ... -lcurses [ library ... ]
#include <curses.h>

int delch(void);
int wdelch(WINDOW *win);
int mvdelch(int y, int x);
int mvwdelch(WINDOW *win, int y, int x);
```

DESCRIPTION With these routines the character under the cursor in the window is deleted; all characters to the right of the cursor on the same line are moved to the left one position and the last character on the line is filled with a blank. The cursor position does not change (after moving to *y, x*, if specified). This does not imply use of the hardware delete character feature.

RETURN VALUES All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), attributes(5)

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `delch()`, `mvdelch()`, and `mvwdelch()` may be macros.

NAME curs_deleteln, deleteln, wdeleteln, insdelln, winsdelln, insertln, winsertln – delete and insert lines in a curses window

SYNOPSIS

```
cc [ flag ... ] file ... -lcurses [ library ... ]
#include <curses.h>

int deleteln(void);
int wdeleteln(WINDOW *win);
int insdelln(int n);
int winsdelln(WINDOW *win, int n);
int insertln(void);
int winsertln(WINDOW *win);
```

DESCRIPTION With the `deleteln()` and `wdeleteln()` routines, the line under the cursor in the window is deleted; all lines below the current line are moved up one line. The bottom line of the window is cleared. The cursor position does not change. This does not imply use of a hardware delete line feature.

With the `insdelln()` and `winsdelln()` routines, for positive *n*, insert *n* lines into the specified window above the current line. The *n* bottom lines are lost. For negative *n*, delete *n* lines (starting with the one under the cursor), and move the remaining lines up. The bottom *n* lines are cleared. The current cursor position remains the same.

With the `insertln()` and `winsertln()` routines, a blank line is inserted above the current line and the bottom line is lost. This does not imply use of a hardware insert line feature.

RETURN VALUES All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that all but `winsdelln()` may be macros.

curses(3CURSES)

| | |
|--------------------|---|
| NAME | curses – CRT screen handling and optimization package |
| SYNOPSIS | <pre>cc [<i>flag...</i>] <i>file...</i> -lcurses [<i>library...</i>] #include <curses.h></pre> |
| DESCRIPTION | <p>The curses library routines give the user a terminal-independent method of updating character screens with reasonable optimization.</p> <p>The curses package allows: overall screen, window and pad manipulation; output to windows and pads; reading terminal input; control over terminal and curses input and output options; environment query routines; color manipulation; use of soft label keys; terminfo access; and access to low-level curses routines.</p> <p>To initialize the routines, the routine <code>initscr()</code> or <code>newterm()</code> must be called before any of the other routines that deal with windows and screens are used. The routine <code>endwin()</code> must be called before exiting. To get character-at-a-time input without echoing (most interactive, screen oriented programs want this), the following sequence should be used:</p> <pre>initscr,cbreak,noecho;</pre> <p>Most programs would additionally use the sequence:</p> <pre>nonl,intrflush(stdscr,FALSE),keypad(stdscr,TRUE);</pre> <p>Before a curses program is run, the tab stops of the terminal should be set and its initialization strings, if defined, must be output. This can be done by executing the <code>tput init</code> command after the shell environment variable <code>TERM</code> has been exported. (See <code>terminfo(4)</code> for further details.)</p> <p>The curses library permits manipulation of data structures, called <i>windows</i>, which can be thought of as two-dimensional arrays of characters representing all or part of a CRT screen. A default window called <code>stdscr</code>, which is the size of the terminal screen, is supplied. Others may be created with <code>newwin(3CURSES)</code>.</p> <p>Windows are referred to by variables declared as <code>WINDOW *</code>. These data structures are manipulated with routines described on 3CURSES pages (whose names begin "curs_"). Among which the most basic routines are <code>move(3CURSES)</code> and <code>addch(3CURSES)</code>. More general versions of these routines are included with names beginning with <code>w</code>, allowing the user to specify a window. The routines not beginning with <code>w</code> affect <code>stdscr</code>.</p> <p>After using routines to manipulate a window, <code>refresh(3CURSES)</code> is called, telling curses to make the user's CRT screen look like <code>stdscr</code>. The characters in a window are actually of type <code>chtype</code>, (character and attribute data) so that other information about the character may also be stored with each character.</p> |

Special windows called *pads* may also be manipulated. These are windows which are not constrained to the size of the screen and whose contents need not be completely displayed. See `curs_pad(3CURSES)` for more information.

In addition to drawing characters on the screen, video attributes and colors may be included, causing the characters to show up in such modes as underlined, in reverse video, or in color on terminals that support such display enhancements. Line drawing characters may be specified to be output. On input, `curses` is also able to translate arrow and function keys that transmit escape sequences into single values. The video attributes, line drawing characters, and input values use names, defined in `<curses.h>`, such as `A_REVERSE`, `ACS_HLINE`, and `KEY_LEFT`.

If the environment variables `LINES` and `COLUMNS` are set, or if the program is executing in a window environment, line and column information in the environment will override information read by `terminfo`. This would effect a program running in an AT&T 630 layer, for example, where the size of a screen is changeable.

If the environment variable `TERMINFO` is defined, any program using `curses` checks for a local terminal definition before checking in the standard place. For example, if `TERM` is set to `att4424`, then the compiled terminal definition is found in

```
/usr/share/lib/terminfo/a/att4424.
```

(The 'a' is copied from the first letter of `att4424` to avoid creation of huge directories.) However, if `TERMINFO` is set to `$HOME/myterms`, `curses` first checks

```
$HOME/myterms/a/att4424,
```

and if that fails, it then checks

```
/usr/share/lib/terminfo/a/att4424.
```

This is useful for developing experimental definitions or when write permission in `/usr/share/lib/terminfo` is not available.

The integer variables `LINES` and `COLS` are defined in `<curses.h>` and will be filled in by `initscr` with the size of the screen. The constants `TRUE` and `FALSE` have the values 1 and 0, respectively.

The `curses` routines also define the `WINDOW *` variable `curscr` which is used for certain low-level operations like clearing and redrawing a screen containing garbage. The `curscr` can be used in only a few routines.

International Functions

The number of bytes and the number of columns to hold a character from the supplementary character set is locale-specific (locale category `LC_CTYPE`) and can be specified in the character class table.

curses(3CURSES)

For editing, operating at the character level is entirely appropriate. For screen formatting, arbitrary movement of characters on screen is not desirable.

Overwriting characters (`addch`, for example) operates on a screen level. Overwriting a character by a character that requires a different number of columns may produce *orphaned columns*. These orphaned columns are filled with background characters.

Inserting characters (`insch`, for example) operates on a character level (that is, at the character boundaries). The specified character is inserted right before the character, regardless of which column of a character the cursor points to. Before insertion, the cursor position is adjusted to the first column of the character.

As with inserting characters, deleting characters (`delch`, for example) operates on a character level (that is, at the character boundaries). The character at the cursor is deleted whichever column of the character the cursor points to. Before deletion, the cursor position is adjusted to the first column of the character.

A *multi-column* character cannot be put on the last column of a line. When such attempts are made, the last column is set to the background character. In addition, when such an operation creates orphaned columns, the orphaned columns are filled with background characters.

Overlapping and overwriting a window follows the operation of overwriting characters around its edge. The orphaned columns, if any, are handled as in the character operations.

The cursor is allowed to be placed anywhere in a window. If the insertion or deletion is made when the cursor points to the second or later column position of a character that holds multiple columns, the cursor is adjusted to the first column of the character before the insertion or deletion.

**Routine and
Argument Names**

Many `curses` routines have two or more versions. The routines prefixed with `w` require a window argument. The routines prefixed with `p` require a pad argument. Those without a prefix generally use `stdscr`.

The routines prefixed with `mv` require an x and y coordinate to move to before performing the appropriate action. The `mv` routines imply a call to `move(3CURSES)` before the call to the other routine. The coordinate y always refers to the row (of the window), and x always refers to the column. The upper left-hand corner is always (0,0), not (1,1).

The routines prefixed with `mvw` take both a window argument and x and y coordinates. The window argument is always specified before the coordinates.

In each case, *win* is the window affected, and *pad* is the pad affected; *win* and *pad* are always pointers to type `WINDOW`

Option setting routines require a Boolean flag *bf* with the value TRUE or FALSE; *bf* is always of type `bool`. The variables *ch* and *attrs* below are always of type `chtype`. The types `WINDOW`, `SCREEN`, `bool`, and `chtype` are defined in `< curses . h >`. The type `TERMINAL` is defined in `< term . h >`. All other arguments are integers.

**Routine Name
Index**

The following table lists each `curses` routine and the name of the manual page on which it is described.

| curses Routine Name | Manual Page Name |
|-------------------------------|---|
| <code>addch</code> | <code> curs_ addch(3CURSES)</code> |
| <code>addchnstr</code> | <code> curs_ addchnstr(3CURSES)</code> |
| <code>addchstr</code> | <code> curs_ addchstr(3CURSES)</code> |
| <code>addnstr</code> | <code> curs_ addnstr(3CURSES)</code> |
| <code>addnwstr</code> | <code> curs_ addnwstr(3CURSES)</code> |
| <code>addstr</code> | <code> curs_ addstr(3CURSES)</code> |
| <code>addwch</code> | <code> curs_ addwch(3CURSES)</code> |
| <code>addwchnstr</code> | <code> curs_ addwchnstr(3CURSES)</code> |
| <code>addwchstr</code> | <code> curs_ addwchstr(3CURSES)</code> |
| <code>addwstr</code> | <code> curs_ addwstr(3CURSES)</code> |
| <code>adjcurspos</code> | <code> curs_ alecompat(3CURSES)</code> |
| <code>attroff</code> | <code> curs_ attr(3CURSES)</code> |
| <code>attron</code> | <code> curs_ attr(3CURSES)</code> |
| <code>attrset</code> | <code> curs_ attr(3CURSES)</code> |
| <code>baudrate</code> | <code> curs_ termattrs(3CURSES)</code> |
| <code>beep</code> | <code> curs_ beep(3CURSES)</code> |
| <code>bkgd</code> | <code> curs_ bkgd(3CURSES)</code> |
| <code>bkgdset</code> | <code> curs_ bkgd(3CURSES)</code> |
| <code>border</code> | <code> curs_ border(3CURSES)</code> |
| <code>box</code> | <code> curs_ border(3CURSES)</code> |
| <code>can_change_color</code> | <code> curs_ color(3CURSES)</code> |
| <code>cbreak</code> | <code> curs_ inopts(3CURSES)</code> |
| <code>clear</code> | <code> curs_ clear(3CURSES)</code> |
| <code>clearok</code> | <code> curs_ outopts(3CURSES)</code> |
| <code>clrbot</code> | <code> curs_ clear(3CURSES)</code> |

curses(3CURSES)

| | |
|----------------|-------------------------|
| clrtoeol | curs_clear(3CURSES) |
| color_content | curs_color(3CURSES) |
| copywin | curs_overlay(3CURSES) |
| curs_set | curs_kernel(3CURSES) |
| def_prog_mode | curs_kernel(3CURSES) |
| def_shell_mode | curs_kernel(3CURSES) |
| del_curterm | curs_terminfo(3CURSES) |
| delay_output | curs_util(3CURSES) |
| delch | curs_delch(3CURSES) |
| deleteln | curs_deleteln(3CURSES) |
| delscreen | curs_initscr(3CURSES) |
| delwin | curs_window(3CURSES) |
| derwin | curs_window(3CURSES) |
| doupdate | curs_refresh(3CURSES) |
| dupwin | curs_window(3CURSES) |
| echo | curs_inopts(3CURSES) |
| echochar | curs_addch(3CURSES) |
| echowchar | curs_addwch(3CURSES) |
| endwin | curs_initscr(3CURSES) |
| erase | curs_clear(3CURSES) |
| erasechar | curs_termattrs(3CURSES) |
| filter | curs_util(3CURSES) |
| flash | curs_beep(3CURSES) |
| flushinp | curs_util(3CURSES) |
| getbegyx | curs_getyx(3CURSES) |
| getch | curs_getch(3CURSES) |
| getmaxyx | curs_getyx(3CURSES) |
| getnwstr | curs_getwstr(3CURSES) |
| getparyx | curs_getyx(3CURSES) |
| getstr | curs_getstr(3CURSES) |
| getsyx | curs_kernel(3CURSES) |

curses(3CURSES)

| | |
|------------|-------------------------|
| getwch | curs_getwch(3CURSES) |
| getwin | curs_util(3CURSES) |
| getwstr | curs_getwstr(3CURSES) |
| getyx | curs_getyx(3CURSES) |
| halfdelay | curs_inopts(3CURSES) |
| has_colors | curs_color(3CURSES) |
| has_ic | curs_termattrs(3CURSES) |
| has_il | curs_termattrs(3CURSES) |
| idcok | curs_outopts(3CURSES) |
| idlok | curs_outopts(3CURSES) |
| immedok | curs_outopts(3CURSES) |
| inch | curs_inch(3CURSES) |
| inchstr | curs_inchstr(3CURSES) |
| inchstr | curs_inchstr(3CURSES) |
| init_color | curs_color(3CURSES) |
| init_pair | curs_color(3CURSES) |
| initscr | curs_initscr(3CURSES) |
| innstr | curs_instr(3CURSES) |
| innwstr | curs_inwstr(3CURSES) |
| insch | curs_insch(3CURSES) |
| insdelln | curs_deleteln(3CURSES) |
| insertln | curs_deleteln(3CURSES) |
| insnstr | curs_insstr(3CURSES) |
| insnwstr | curs_inswstr(3CURSES) |
| insstr | curs_insstr(3CURSES) |
| instr | curs_instr(3CURSES) |
| inswch | curs_inswch(3CURSES) |
| inswstr | curs_inswstr(3CURSES) |
| intrflush | curs_inopts(3CURSES) |
| inwch | curs_inwch(3CURSES) |
| inwchnstr | curs_inwchstr(3CURSES) |

curses(3CURSES)

| | |
|----------------|-------------------------|
| inwchstr | curs_inwchstr(3CURSES) |
| inwstr | curs_inwstr(3CURSES) |
| is_linetouched | curs_touch(3CURSES) |
| is_wintouched | curs_touch(3CURSES) |
| isendwin | curs_initscr(3CURSES) |
| keyname | curs_util(3CURSES) |
| keypad | curs_inopts(3CURSES) |
| killchar | curs_termattrs(3CURSES) |
| leaveok | curs_outopts(3CURSES) |
| longname | curs_termattrs(3CURSES) |
| meta | curs_inopts(3CURSES) |
| move | curs_move(3CURSES) |
| movenextch | curs_alecompat(3CURSES) |
| moveprevch | curs_alecompat(3CURSES) |
| mvaddch | curs_addch(3CURSES) |
| mvaddchnstr | curs_addchstr(3CURSES) |
| mvaddchstr | curs_addchstr(3CURSES) |
| mvaddnstr | curs_addstr(3CURSES) |
| mvaddnwstr | curs_addwstr(3CURSES) |
| mvaddstr | curs_addstr(3CURSES) |
| mvaddwch | curs_addwch(3CURSES) |
| mvaddwchnstr | curs_addwchstr(3CURSES) |
| mvaddwchstr | curs_addwchstr(3CURSES) |
| mvaddwstr | curs_addwstr(3CURSES) |
| mvcur | curs_terminfo(3CURSES) |
| mvdelch | curs_delch(3CURSES) |
| mvderwin | curs_window(3CURSES) |
| mvgetch | curs_getch(3CURSES) |
| mvgetnwstr | curs_getwstr(3CURSES) |
| mvgetstr | curs_getstr(3CURSES) |
| mvgetwch | curs_getwch(3CURSES) |

curses(3CURSES)

| | |
|---------------|-------------------------|
| mvgetwstr | curs_getwstr(3CURSES) |
| mvinch | curs_inch(3CURSES) |
| mvinchnstr | curs_inchstr(3CURSES) |
| mvinchstr | curs_inchstr(3CURSES) |
| mvinnstr | curs_instr(3CURSES) |
| mvinnwstr | curs_inwstr(3CURSES) |
| mvinsch | curs_insch(3CURSES) |
| mvinsnstr | curs_insstr(3CURSES) |
| mvinswstr | curs_inswstr(3CURSES) |
| mvinsstr | curs_insstr(3CURSES) |
| mvinstr | curs_instr(3CURSES) |
| mvinswch | curs_inswch(3CURSES) |
| mvinswstr | curs_inswstr(3CURSES) |
| mvinwch | curs_inwch(3CURSES) |
| mvinwchnstr | curs_inwchstr(3CURSES) |
| mvinwchstr | curs_inwchstr(3CURSES) |
| mvinwstr | curs_inwstr(3CURSES) |
| mvprintw | curs_printw(3CURSES) |
| mvscanw | curs_scanw(3CURSES) |
| mvwaddch | curs_addch(3CURSES) |
| mvwaddchnstr | curs_addchstr(3CURSES) |
| mvwaddchstr | curs_addchstr(3CURSES) |
| mvwaddnstr | curs_addstr(3CURSES) |
| mvwaddnwstr | curs_addwstr(3CURSES) |
| mvwaddstr | curs_addstr(3CURSES) |
| mvwaddwch | curs_addwch(3CURSES) |
| mvwaddwchnstr | curs_addwchstr(3CURSES) |
| mvwaddwchstr | curs_addwchstr(3CURSES) |
| mvwaddwstr | curs_addwstr(3CURSES) |
| mvwdelch | curs_delch(3CURSES) |
| mvwgetch | curs_getch(3CURSES) |

curses(3CURSES)

| | |
|--------------|------------------------|
| mvwgetnwstr | curs_getwstr(3CURSES) |
| mvwgetstr | curs_getstr(3CURSES) |
| mvwgetwch | curs_getwch(3CURSES) |
| mvwgetwstr | curs_getwstr(3CURSES) |
| mvwin | curs_window(3CURSES) |
| mvwinch | curs_inch(3CURSES) |
| mvwinchnstr | curs_inchstr(3CURSES) |
| mvwinchstr | curs_inchstr(3CURSES) |
| mvwinnstr | curs_instr(3CURSES) |
| mvwinnwstr | curs_inwstr(3CURSES) |
| mvwinsch | curs_insch(3CURSES) |
| mvwinsnstr | curs_insstr(3CURSES) |
| mvwinsstr | curs_insstr(3CURSES) |
| mvwinstr | curs_instr(3CURSES) |
| mvwinswch | curs_inswch(3CURSES) |
| mvwinswstr | curs_inswstr(3CURSES) |
| mvwinwch | curs_inwch(3CURSES) |
| mvwinwchnstr | curs_inwchstr(3CURSES) |
| mvwinwchstr | curs_inwchstr(3CURSES) |
| mvwinwstr | curs_inwstr(3CURSES) |
| mvwprintw | curs_printw(3CURSES) |
| mvwscanw | curs_scanw(3CURSES) |
| napms | curs_kernel(3CURSES) |
| newpad | curs_pad(3CURSES) |
| newterm | curs_initscr(3CURSES) |
| newwin | curs_window(3CURSES) |
| nl | curs_outopts(3CURSES) |
| nocbreak | curs_inopts(3CURSES) |
| nodelay | curs_inopts(3CURSES) |
| noecho | curs_inopts(3CURSES) |
| nonl | curs_outopts(3CURSES) |

curses(3CURSES)

| | |
|------------------|------------------------|
| noqiflush | curs_inopts(3CURSES) |
| noraw | curs_inopts(3CURSES) |
| notimeout | curs_inopts(3CURSES) |
| overlay | curs_overlay(3CURSES) |
| overwrite | curs_overlay(3CURSES) |
| pair_content | curs_color(3CURSES) |
| pechochar | curs_pad(3CURSES) |
| pechowchar | curs_pad(3CURSES) |
| pnoutrefresh | curs_pad(3CURSES) |
| prefresh | curs_pad(3CURSES) |
| printw | curs_printw(3CURSES) |
| putp | curs_terminfo(3CURSES) |
| putwin | curs_util(3CURSES) |
| qiflush | curs_inopts(3CURSES) |
| raw | curs_inopts(3CURSES) |
| redrawwin | curs_refresh(3CURSES) |
| refresh | curs_refresh(3CURSES) |
| reset_prog_mode | curs_kernel(3CURSES) |
| reset_shell_mode | curs_kernel(3CURSES) |
| resetty | curs_kernel(3CURSES) |
| restartterm | curs_terminfo(3CURSES) |
| ripline | curs_kernel(3CURSES) |
| savetty | curs_kernel(3CURSES) |
| scanw | curs_scanw(3CURSES) |
| scr_dump | curs_scr_dump(3CURSES) |
| scr_init | curs_scr_dump(3CURSES) |
| scr_restore | curs_scr_dump(3CURSES) |
| scr_set | curs_scr_dump(3CURSES) |
| scroll | curs_scroll(3CURSES) |
| scrollok | curs_outopts(3CURSES) |
| set_curterm | curs_terminfo(3CURSES) |

curses(3CURSES)

| | |
|-----------------|-------------------------|
| set_term | curs_initscr(3CURSES) |
| setscreg | curs_outopts(3CURSES) |
| setsyx | curs_kernel(3CURSES) |
| setterm | curs_terminfo(3CURSES) |
| setupterm | curs_terminfo(3CURSES) |
| slk_attroff | curs_slk(3CURSES) |
| slk_attron | curs_slk(3CURSES) |
| slk_attrset | curs_slk(3CURSES) |
| slk_clear | curs_slk(3CURSES) |
| slk_init | curs_slk(3CURSES) |
| slk_label | curs_slk(3CURSES) |
| slk_noutrefresh | curs_slk(3CURSES) |
| slk_refresh | curs_slk(3CURSES) |
| slk_restore | curs_slk(3CURSES) |
| slk_set | curs_slk(3CURSES) |
| slk_touch | curs_slk(3CURSES) |
| srl | curs_scroll(3CURSES) |
| standend | curs_attr(3CURSES) |
| standout | curs_attr(3CURSES) |
| start_color | curs_color(3CURSES) |
| subpad | curs_pad(3CURSES) |
| subwin | curs_window(3CURSES) |
| syncok | curs_window(3CURSES) |
| termattrs | curs_termattrs(3CURSES) |
| termname | curs_termattrs(3CURSES) |
| tgetent | curs_termcap(3CURSES) |
| tgetflag | curs_termcap(3CURSES) |
| tgetnum | curs_termcap(3CURSES) |
| tgetstr | curs_termcap(3CURSES) |
| tgoto | curs_termcap(3CURSES) |
| tigetflag | curs_terminfo(3CURSES) |

curses(3CURSES)

| | |
|-------------|-------------------------|
| tigetnum | curs_terminfo(3CURSES) |
| tigetstr | curs_terminfo(3CURSES) |
| timeout | curs_inopts(3CURSES) |
| touchline | curs_touch(3CURSES) |
| touchwin | curs_touch(3CURSES) |
| tparm | curs_terminfo(3CURSES) |
| tputs | curs_terminfo(3CURSES) |
| typeahead | curs_inopts(3CURSES) |
| unctrl | curs_util(3CURSES) |
| ungetch | curs_getch(3CURSES) |
| ungetwch | curs_getwch(3CURSES) |
| untouchwin | curs_touch(3CURSES) |
| use_env | curs_util(3CURSES) |
| vidattr | curs_terminfo(3CURSES) |
| vidputs | curs_terminfo(3CURSES) |
| vwprintw | curs_printw(3CURSES) |
| vwscanw | curs_scanw(3CURSES) |
| waddch | curs_addch(3CURSES) |
| waddchnstr | curs_addchstr(3CURSES) |
| waddchstr | curs_addchstr(3CURSES) |
| waddnstr | curs_addstr(3CURSES) |
| waddnwstr | curs_addwstr(3CURSES) |
| waddstr | curs_addstr(3CURSES) |
| waddwch | curs_addwch(3CURSES) |
| waddwchnstr | curs_addwchstr(3CURSES) |
| waddwchstr | curs_addwchstr(3CURSES) |
| waddwstr | curs_addwstr(3CURSES) |
| wadjcurspos | curs_alecompat(3CURSES) |
| wattroff | curs_attr(3CURSES) |
| wattron | curs_attr(3CURSES) |
| wattrset | curs_attr(3CURSES) |

curses(3CURSES)

| | |
|------------|------------------------|
| wbkgd | curs_bkgd(3CURSES) |
| wbkgdset | curs_bkgd(3CURSES) |
| wborder | curs_border(3CURSES) |
| wclear | curs_clear(3CURSES) |
| wclrtoBOT | curs_clear(3CURSES) |
| wclrtoEOL | curs_clear(3CURSES) |
| wcursyncup | curs_window(3CURSES) |
| wdelch | curs_delch(3CURSES) |
| wdeleteln | curs_deleteln(3CURSES) |
| wechochar | curs_addch(3CURSES) |
| wechowchar | curs_addwch(3CURSES) |
| werase | curs_clear(3CURSES) |
| wgetch | curs_getch(3CURSES) |
| wgetnstr | curs_getstr(3CURSES) |
| wgetnwstr | curs_getwstr(3CURSES) |
| wgetstr | curs_getstr(3CURSES) |
| wgetwch | curs_getwch(3CURSES) |
| wgetwstr | curs_getwstr(3CURSES) |
| whline | curs_border(3CURSES) |
| winch | curs_inch(3CURSES) |
| winchnstr | curs_inchstr(3CURSES) |
| winchstr | curs_inchstr(3CURSES) |
| winnstr | curs_instr(3CURSES) |
| winnwstr | curs_inwstr(3CURSES) |
| winsch | curs_insch(3CURSES) |
| winsdelln | curs_deleteln(3CURSES) |
| winsertln | curs_deleteln(3CURSES) |
| winsnstr | curs_insstr(3CURSES) |
| winsnwstr | curs_inswstr(3CURSES) |
| winsstr | curs_insstr(3CURSES) |
| winstr | curs_instr(3CURSES) |

| | |
|--------------|-------------------------|
| winswch | curs_inswch(3CURSES) |
| winswstr | curs_inswstr(3CURSES) |
| winwch | curs_inwch(3CURSES) |
| winwchnstr | curs_inwchstr(3CURSES) |
| winwchstr | curs_inwchstr(3CURSES) |
| winwstr | curs_inwstr(3CURSES) |
| wmove | curs_move(3CURSES) |
| wmovenextch | curs_alecompat(3CURSES) |
| wmoveprevch | curs_alecompat(3CURSES) |
| wnoutrefresh | curs_refresh(3CURSES) |
| wprintw | curs_printw(3CURSES) |
| wredrawln | curs_refresh(3CURSES) |
| wrefresh | curs_refresh(3CURSES) |
| wscanw | curs_scanw(3CURSES) |
| wscrl | curs_scroll(3CURSES) |
| wsetscrreg | curs_outopts(3CURSES) |
| wstandend | curs_attr(3CURSES) |
| wstandout | curs_attr(3CURSES) |
| wsyncdown | curs_window(3CURSES) |
| wsyncup | curs_window(3CURSES) |
| wtimeout | curs_inopts(3CURSES) |
| wtouchln | curs_touch(3CURSES) |
| wvline | curs_border(3CURSES) |

RETURN VALUES

Routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the routine descriptions.

All macros return the value of the `w` version, except `setscrreg()`, `wsetscrreg()`, `getyx()`, `getbegyx()`, and `getmaxyx()`. The return values of `setscrreg()`, `wsetscrreg()`, `getyx()`, `getbegyx()`, and `getmaxyx()` are undefined (that is, these should not be used as the right-hand side of assignment statements).

Routines that return pointers return `NULL` on error.

curses(3CURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3XCURSES)`, `libcurses(3LIB)`, `libcurses(3XCURSES)`, `terminfo(4)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

| | |
|--------------------|--|
| NAME | curses – introduction and overview of X/Open Curses |
| DESCRIPTION | <p>The Curses screen management package conforms fully with Issue 4, Version 2 of the X/Open Curses specification. It provides a set of internationalized functions and macros for creating and modifying input and output to a terminal screen. This includes functions for creating windows, highlighting text, writing to the screen, reading from user input, and moving the cursor.</p> <p>X/Open Curses is a terminal-independent package, providing a common user interface to a variety of terminal types. Its portability is facilitated by the Terminfo database which contains a compiled definition of each terminal type. By referring to the database information X/Open Curses gains access to low-level details about individual terminals.</p> <p>X/Open Curses tailors its activities to the terminal type specified by the <code>TERM</code> environment variable. The <code>TERM</code> environment variable may be set in the Korn Shell (see <code>ksh(1)</code>) by typing:</p> <pre>export TERM=terminal_name</pre> <p>To set environment variables using other command line interfaces or shells, see the <code>environ(5)</code> manual page.</p> <p>Three additional environment variables are useful, and can be set in the Korn Shell:</p> <ol style="list-style-type: none"> 1. If you have an alternate Terminfo database containing terminal types that are not available in the system default database <code>/usr/share/lib/terminfo</code>, you can specify the <code>TERMINFO</code> environment variable to point to this alternate database: <pre>export TERMINFO=path</pre> <p>This <i>path</i> specifies the location of the alternate compiled Terminfo database whose structure consists of directory names 0 to 9 and a to z (which represent the first letter of the compiled terminal definition file name). The alternate database specified by <code>TERMINFO</code> is examined before the system default database. If the terminal type specified by <code>TERM</code> cannot be found in either database, the default terminal type <i>dumb</i> is assumed.</p> 2. To specify a window width smaller than your screen width (for example, in situations where your communications line is slow), set the <code>COLUMNS</code> environment variable to the number of vertical columns you want between the left and right margins: <pre>export COLUMNS=number</pre> <p>The <i>number</i> of columns may be set to a number smaller than the screen size; however, if set larger than the screen or window width, the results are undefined. The value set using this environment variable takes precedence over the value normally used for the terminal.</p> 3. To specify a window height smaller than your current screen height (for example, in situations where your communications line is slow), override the <code>LINES</code> environment variable by setting it to a smaller number of horizontal lines: |

curses(3XCURSES)

```
export LINES=number
```

The *number* of lines may be set to a number smaller than the screen height; however, if set larger than the screen or window height, the results are undefined. The value set using this environment variable takes precedence over the value normally used for the terminal.

Data Types

X/Open Curses defines the following data types:

| | |
|-----------------------|---|
| <code>attr_t</code> | An integral type that holds an OR-ed set of attributes. The attributes acceptable are those which begin with the <code>WA_</code> prefix . |
| <code>bool</code> | Boolean data type. |
| <code>cchar_t</code> | A type that refers to a string consisting of a spacing wide character, up to 5 non-spacing wide characters, and zero or more attributes of any type. See Attributes, Color Pairs, and Renditions. A null <code>cchar_t</code> object terminates arrays of <code>cchar_t</code> objects. |
| <code>chtype</code> | An integral type whose values are formed by OR-ing an "unsigned char" with a color pair. and with zero or more attributes. The attributes acceptable are those which begin with the <code>A_</code> prefix and <code>COLOR_PAIR(3XCURSES)</code> |
| <code>SCREEN</code> | An opaque data type associated with a terminal's display screen. |
| <code>TERMINAL</code> | An opaque data type associated with a terminal. It contains information about the terminal's capabilities (as defined by <code>terminfo</code>), the terminal modes, and current state of input/output operations. |
| <code>wchar_t</code> | An integral data type whose values represent wide characters. |
| <code>WINDOW</code> | An opaque data type associated with a window. |

Screens, Windows, and Terminals

The X/Open Curses manual pages refer at various points to screens, windows (also subwindows, derived windows, and pads), and terminals. The following list defines each of these terms.

| | |
|--------|---|
| Screen | A screen is a terminal's physical output device. The <code>SCREEN</code> data type is associated with a terminal. |
| Window | Window objects are two-dimensional arrays of characters and their renditions. X/Open Curses provides <code>stdscr</code> , a default window which is the size of of the terminal screen. You can use the <code>newwin(3XCURSES)</code> function to create others. |

To refer to a window, use a variable declared as `WINDOW *`. X/Open Curses includes both functions that modify `stdscr`, and more general versions that let you specify a window.

There are three sub-types of windows:

| | |
|----------------|---|
| Subwindow | A window which has been created within another window (the parent window) and whose position has been specified with absolute screen coordinates. The <code>derwin(3XCURSES)</code> and <code>subwin(3XCURSES)</code> functions can be used to create subwindows. |
| Derived Window | A subwindow whose position is defined relative to the parent window's coordinates rather than in absolute terms. |
| Pad | A special type of window that can be larger than the screen. For more information, see the <code>newpad(3XCURSES)</code> man page. |
| Terminal | A terminal is the input and output device which character-based applications use to interact with the user. The <code>TERMINAL</code> data type is associated with such a device. |

Attributes, Color Pairs, and Renditions

A character's rendition consists of its attributes (such as underlining or reverse video) and its color pair (the foreground and background colors). When using `waddstr(3XCURSES)`, `waddchstr(3XCURSES)`, `wprintw(3XCURSES)`, `winsch(3XCURSES)`, and so on, the window's rendition is combined with that character's renditions. The window rendition is the attributes and color set using the `attroff(3XCURSES)` and `attr_off(3XCURSES)` sets of functions. The window's background character and rendition are set with the `bkgdset(3XCURSES)` and `bkgdset(3XCURSES)` sets of functions.

When spaces are written to the screen, the background character and window rendition replace the space. For example, if the background rendition and character is `A_UNDERLINE | '*'`, text written to the window appears underlined and the spaces appear as underlined asterisks.

Each character written retains the rendition that it has obtained. This allows the character to be copied "as is" to or from a window with the `addchstr(3XCURSES)` or `inch(3XCURSES)` functions.

A_ Constant Values for Attributes

You can specify Attributes, Color Pairs, and Renditions attributes using the constants listed in the tables below. The following constants modify objects of type `chtype`:

| Constant | Description |
|---------------------------|--------------------------------|
| <code>A_ALTCHARSET</code> | Alternate character set |
| <code>A_ATTRIBUTES</code> | Bit-mask to extract attributes |
| <code>A_BLINK</code> | Blinking |

| Constant | Description |
|-------------|--|
| A_BOLD | Bold |
| A_CHARTEXT | Bit-mask to extract a character |
| A_COLOR | Bit-mask to extract color-pair information |
| A_DIM | Half-bright |
| A_INVIS | Invisible |
| A_PROTECT | Protected |
| A_REVERSE | Reverse video |
| A_STANDOUT | Highlights specific to terminal |
| A_UNDERLINE | Underline |

WA_ Constant Values for Attributes

The following constants modify objects of type `attr_t`:

| Constant | Description |
|---------------|---------------------------------|
| WA_ALTCHARSET | Alternate character set |
| WA_ATTRIBUTES | Attribute mask |
| WA_BLINK | Blinking |
| WA_BOLD | Bold |
| WA_DIM | Half-bright |
| WA_HORIZONTAL | Horizontal highlight |
| WA_INVIS | Invisible |
| WA_LEFT | Left highlist |
| WA_LOW | Low highlist |
| WA_PROTECT | Protected |
| WA_REVERSE | Reverse video |
| WA_RIGHT | Right highlight |
| WA_STANDOUT | Highlights specific to terminal |
| WA_TOP | Top highlight |
| WA_UNDERLINE | Underline |

| Constant | Description |
|-------------|--------------------|
| WA_VERTICAL | Vertical highlight |

Color Macros

Colors always appear in pairs; the foreground color of the character itself and the background color of the field on which it is displayed. The following color macros are defined:

| Macro | Description |
|---------------|-------------|
| COLOR_BLACK | Black |
| COLOR_BLUE | Blue |
| COLOR_GREEN | Green |
| COLOR_CYAN | Cyan |
| COLOR_RED | Red |
| COLOR_MAGENTA | Magenta |
| COLOR_YELLOW | Yellow |
| COLOR_WHITE | White |

Together, a character's attributes and its color pair form the character's rendition. A character's rendition moves with the character during any scrolling or insert/delete operations. If your terminal lacks support for the specified rendition, X/Open Curses may substitute a different rendition.

The `COLOR_PAIR(3XCURSES)` function modifies a `chtype` object. The `PAIR_NUMBER(3XCURSES)` function extracts the color pair from a `chtype` object.

Functions for Modifying a Window's Color

The following functions modify a window's color:

| Function | Description |
|--|--------------------------------|
| <code>attr_set()</code> , <code>wattr_set()</code> | Change the window's rendition. |
| <code>color_set()</code> , <code>wcolor_set()</code> | Set the window's color |

Non-Spacing Characters

When the `wcwidth(3C)` function returns a width of zero for a character, that character is called a non-spacing character. Non-spacing characters can be written to a window. Each non-spacing character is associated with a spacing character (that is, one which

curses(3XCURSES)

does not have a width of zero) and modifies that character. You cannot address a non-spacing character directly. Whenever you perform an X/Open Curses operation on the associated character, you are implicitly addressing the non-spacing character.

Non-spacing characters do not have a rendition. For functions that use wide characters and a rendition, X/Open Curses ignores any rendition specified for non-spacing characters. Multi-column characters have one rendition that applies to all columns spanned.

Complex Characters

The `cchar_t` data type represents a complex character. A complex character may contain a spacing character, its associated non-spacing characters, and its rendition. This implementation of complex characters supports up to 5 non-spacing characters for each spacing character.

When a `cchar_t` object representing a non-spacing complex character is written to the screen, its rendition is not used, but rather it becomes associated with the rendition of the existing character at that location. The `setcchar(3XCURSES)` function initializes an object of type `cchar_t`. The `getcchar(3XCURSES)` function extracts the contents of a `cchar_t` object.

Display Operations

In adding internationalization support to X/Open Curses, every attempt was made to minimize the number of changes to the historical CURSES package. This enables programs written to use the historical implementation of CURSES to use the internationalized version with little or no modification. The following rules apply to the internationalized X/Open Curses package:

- The cursor can be placed anywhere in the window. Window and screen origins are (0,0).
- A multi-column character cannot be displayed in the last column, because the character would appear truncated. Instead, the background character is displayed in the last column and the multi-column character appears at the beginning of the next line. This is called wrapping.

If the original line is the last line in the scroll region and scrolling is enabled, X/Open Curses moves the contents of each line in the region to the previous line. The first line of the region is lost. The last line of the scrolling region contains any wrapped characters. The remainder of that line is filled with the background character. If scrolling is disabled, X/Open Curses truncates any character that would extend past the last column of the screen.

- Overwrites operate on screen columns. If displaying a single-column or multi-column character results in overwriting only a portion of a multi-column character or characters, background characters are displayed in place of the non-overwritten portions.
- Insertions and deletions operate on whole characters. The cursor is moved to the first column of the character prior to performing the operation.

Overlapping Windows

When windows overlap, it may be necessary to overwrite only part of a multi-column character. As mentioned earlier, the non-overwritten portions are replaced with the background character. This results in issues concerning the `overwrite(3XCURSES)`, `overlay(3XCURSES)`, `copywin(3XCURSES)`, `wnoutrefresh(3XCURSES)`, and `wrefresh(3XCURSES)` functions.

Special Characters

Some functions assign special meanings to certain special characters:

Backspace Moves the cursor one column towards the beginning of the line. If the cursor was already at the beginning of the line, it remains there. All subsequent characters are added or inserted at this point.

Carriage Return Moves the cursor to the beginning of the current line. If the cursor was already at the beginning of the line, it remains there. All subsequent characters are added or inserted at this point.

Newline When adding characters, `X/Open Curses` fills the remainder of the line with the background character (effectively truncating the newline) and scrolls the window as described earlier. All subsequent characters are inserted at the start of the new line.

When inserting characters, `X/Open Curses` fills the remainder of the line with the background character (effectively truncating the line), moves the cursor to the beginning of a new line, and scrolls the window as described earlier. All subsequent characters are placed at the start of the new line.

Tab moves subsequent characters to next horizontal tab stop. Default tab stops are set at 0, 8, 16, and so on.

When adding or inserting characters, `X/Open Curses` inserts or adds the background character into each column until the next tab stop is reached. If there are no remaining tab stops on the current line, wrapping and scrolling occur as described earlier.

Control Characters When `X/Open Curses` functions perform special character processing, they convert control characters to the `^X` notation, where `X` is a single-column character (uppercase, if it is a letter) and writes that notation to the window. Functions that retrieve text from the window will retrieve the converted notation not the original.

`X/Open Curses` displays non-printable bytes, that have their high bit set, using the `M-X` meta notation where `X` is the non-printable byte with its high bit turned off.

curses(3XCURSES)

Input Processing

There are four input modes possible with X/Open Curses that affect the behavior of input functions like `getch(3XCURSES)` and `getnstr(3XCURSES)`.

| | |
|-----------------------------|---|
| Line Canonical (Cooked) | <p>In line input mode, the terminal driver handles the input of line units as well as <code>SIGERASE</code> and <code>SIGKILL</code> character processing. See <code>termio(7I)</code> for more information.</p> <p>In this mode, the <code>getch()</code> and <code>getnstr()</code> functions will not return until a complete line has been read by the terminal driver, at which point only the requested number of bytes/characters are returned. The rest of the line unit remains unread until subsequent call to the <code>getch()</code> or <code>getnstr()</code> functions.</p> <p>The functions <code>nocbreak(3XCURSES)</code> and <code>noraw(3XCURSES)</code> are used to enter this mode. These functions are described on the <code>cbreak(3XCURSES)</code> man page which also details which <code>termios</code> flags are enabled.</p> <p>Of the modes available, this one gives applications the least amount of control over input. However, it is the only input mode possible on a block mode terminal.</p> |
| <code>cbreak</code> Mode | <p>Byte/character input provides a finer degree of control. The terminal driver passes each byte read to the application without interpreting erase and kill characters. It is the application's responsibility to handle line editing. It is unknown whether the signal characters (<code>SIGINTR</code>, <code>SIGQUIT</code>, <code>SIGSUSP</code>) and flow control characters (<code>SIGSTART</code>, <code>SIGSTOP</code>) are enabled. To ensure that they are, call the <code>noraw()</code> function first, then call the <code>cbreak()</code> function.</p> |
| <code>halfdelay</code> Mode | <p>This is the same as the <code>cbreak()</code> mode with a timeout. The terminal driver waits for a byte to be received or for a timer to expire, in which case the <code>getch()</code> function either returns a byte or <code>ERR</code> respectively. This mode overrides timeouts set for an individual window with the <code>wtimeout()</code> function.</p> |
| <code>raw</code> Mode | <p>This mode provides byte/character input with the most control for an application. It is similar to <code>cbreak()</code> mode, but also disables signal character processing (<code>SIGINTR</code>, <code>SIGSUSP</code>, <code>SIGQUIT</code>) and flow control processing (<code>SIGSTART</code>, <code>SIGSTOP</code>) so that the application can process them as it wants.</p> |

These modes affect all X/Open Curses input. The default input mode is inherited from the parent process when the application starts up.

A timeout similar to `halfdelay(3XCURSES)` can be applied to individual windows (see `timeout(3XCURSES)`). The `nodelay(3XCURSES)` function is equivalent to setting `wtimeout(3XCURSES)` for a window with a zero timeout (non-blocking) or infinite delay (blocking).

To handle function keys, `keypad(3XCURSES)` must be enabled. When it is enabled, the `getch()` function returns a `KEY_` constant for a uniquely encoded key defined for that terminal. When `keypad()` is disabled, the `getch()` function returns the individual bytes composing the function key (see `getch(3XCURSES)` and `wget_wch(3XCURSES)`). By default, `keypad()` is disabled.

When processing function keys, once the first byte is recognized, a timer is set for each subsequent byte in the sequence. If any byte in the function key sequence is not received before the timer expires, the bytes already received are pushed into a buffer and the original first byte is returned. Subsequent X/Open Curses input would take bytes from the buffer until exhausted, after which new input from the terminal will be requested. Enabling and disabling of the function key interbyte timer is handled by the `notimeout(3XCURSES)` function. By default, `notimeout()` is disabled (that is, the timer is used).

X/Open Curses always disables the terminal driver's echo processing. The `echo(3XCURSES)` and `noecho(3XCURSES)` functions control X/Open Curses software echoing. When software echoing is enabled, X/Open Curses input functions echo printable characters, control keys, and meta keys in the input window at the last cursor position. Functions keys are never echoed. When software echoing is disabled, it is the application's responsibility to handle echoing.

EXAMPLES

EXAMPLE 1 Copying Single-Column Characters Over Single-Column Characters

In the upcoming examples, some characters have special meanings:

- {, [, and (represent the left halves of multi-column characters. },], and) represent the corresponding right halves of the same multi-column characters.
- Alphanumeric characters and periods (.) represent single-column characters.
- The number sign (#) represents the background character.

```
copywin(s, t, 0, 1, 0, 1, 1, 3, 0)
```

| s | t | → | t |
|--------|-------|---|--------|
| abcdef | | | .bcd.. |
| ghijkl | | | .hij.. |

There are no special problems with this situation.

EXAMPLE 2 Copying Multi-column Characters Over Single-Column Characters

```
copywin(s, t, 0, 1, 0, 1, 1, 3, 0)
```

```

      s           t           →    t
a[]def           .....         .[]d..
gh()kl           .....         .h()..

```

There are no special problems with this situation.

EXAMPLE 3 Copying Single-Column Characters From Source Overlaps Multi-column Characters In Target

```
copywin(s, t, 0, 1, 0, 1, 1, 3, 0)
```

```

      s           t           →    t
abcdef           []....         #bcd..
ghijk tol       ...().         .hij#.

```

Overwriting multi-column characters in `t` has resulted in the `#` background characters being required to erase the remaining halves of the target's multi-column characters.

EXAMPLE 4 Copy Incomplete Multi-column Characters From Source To Target.

```
copywin(s, t, 0, 1, 0, 1, 1, 3, 0)
```

```

      s           t           →    t
[]cdef           123456         []cd56
ghi()l           789012         7hi()2

```

The `]` and `(` halves of the multi-column characters have been copied from the source and expanded in the target outside of the specified target region.

Consider a pop-up dialog box that contains single-column characters and a base window that contains multi-column characters and you do the following:

```

save=dupwin(dialog); /* create backing store */
overwrite(cursor, save); /* save region to be overlaid */
wrefresh(dialog); /* display dialog */
wrefresh(save); /* restore screen image */
delwin(save); /* release backing store */

```

You can use code similar to this to implement generic `popup()` and `popdown()` routines in a variety of CURSES implementations (including BSD UNIX, and UNIX System V). In the simple case where the base window contains single-column characters only, it would correctly restore the image that appeared on the screen before the dialog box was displayed.

However, with multi-column characters, the `overwrite()` function might save a region with incomplete multi-column characters. The `wrefresh(dialog)` statement results in the behavior described in example 3 above. The behavior described in this example (that is, example 4) allows the `wrefresh(save)` statement to restore the window correctly.

EXAMPLE 5 Copying An Incomplete Multi-column Character To Region Next To Screen Margin (Not A Window Edge)

Two cases of copying an incomplete multi-column character to a region next to a screen margin follow:

```
copywin(s, t, 0, 1, 0, 0, 1, 2, 0)
```

```

      s           t           →    t
      []cdef      123456      #cd456
      ghijkl      789012      hij012
    
```

The background character (#) replaces the] character that would have been copied from the source, because it is not possible to expand the multi-column character to its complete form.

```
copywin(s, t, 0, 1, 0, 3, 1, 5, 0)
```

```

      s           t           →    t
      abcdef      123456      123bcd
      ghi()l      789012      789hi#
    
```

This second example is the same as the first, but with the right margin.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO

`ksh(1)`, `COLOR_PAIR(3XCURSES)`, `PAIR_NUMBER(3XCURSES)`, `addchstr(3XCURSES)`, `attr_off(3XCURSES)`, `attroff(3XCURSES)`, `bkgdset(3XCURSES)`, `bkgrndset(3XCURSES)`, `cbreak(3XCURSES)`, `copywin(3XCURSES)`, `derwin(3XCURSES)`, `echo(3XCURSES)`, `getcchar(3XCURSES)`, `getch(3XCURSES)`, `getnstr(3XCURSES)`, `halfdelay(3XCURSES)`, `inch(3XCURSES)`, `keypad(3XCURSES)`, `libcurses(3XCURSES)`, `newpad(3XCURSES)`, `newwin(3XCURSES)`, `nocbreak(3XCURSES)`, `nodelay(3XCURSES)`, `noecho(3XCURSES)`, `noraw(3XCURSES)`, `notimeout(3XCURSES)`, `overlay(3XCURSES)`, `overwrite(3XCURSES)`, `setcchar(3XCURSES)`, `subwin(3XCURSES)`, `timeout(3XCURSES)`, `waddchstr(3XCURSES)`, `waddstr(3XCURSES)`, `wcwidth(3C)`, `wget_wch(3XCURSES)`, `winsch(3XCURSES)`, `wnoutrefresh(3XCURSES)`, `wprintw(3XCURSES)`, `wrefresh(3XCURSES)`, `wtimeout(3XCURSES)`, `attributes(5)`, `environ(5)`, `standards(5)`, `termio(7I)`

curs_getch(3CURSES)

| NAME | <code>curs_getch</code> , <code>getch</code> , <code>wgetch</code> , <code>mvgetch</code> , <code>mvwgetch</code> , <code>ungetch</code> – get (or push back) characters from curses terminal keyboard | | | | |
|------------------------|--|------|----------|------------------------|-----------|
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int getch(void); int wgetch(WINDOW *<i>win</i>); int mvgetch(int <i>y</i>, int <i>x</i>); int mvwgetch(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>); int ungetch(int <i>ch</i>);</pre> | | | | |
| DESCRIPTION | <p>With the <code>getch()</code>, <code>wgetch()</code>, <code>mvgetch()</code>, and <code>mvwgetch()</code> routines a character is read from the terminal associated with the window. In no-delay mode, if no input is waiting, the value <code>ERR</code> is returned. In delay mode, the program waits until the system passes text through to the program. Depending on the setting of <code>cbreak()</code>, this is after one character (<code>cbreak</code> mode), or after the first newline (<code>nocbreak</code> mode). In half-delay mode, the program waits until a character is typed or the specified timeout has been reached. Unless <code>noecho()</code> has been set, the character will also be echoed into the designated window.</p> <p>If the window is not a pad, and it has been moved or modified since the last call to <code>wrefresh()</code>, <code>wrefresh()</code> will be called before another character is read.</p> <p>If <code>keypad()</code> is <code>TRUE</code>, and a function key is pressed, the token for that function key is returned instead of the raw characters. Possible function keys are defined in <code><curses.h></code> with integers beginning with <code>0401</code>, whose names begin with <code>KEY_</code>. If a character that could be the beginning of a function key (such as escape) is received, <code>curses</code> sets a timer. If the remainder of the sequence does not come in within the designated time, the character is passed through; otherwise, the function key value is returned. For this reason, many terminals experience a delay between the time a user presses the escape key and the escape is returned to the program. Since tokens returned by these routines are outside the ASCII range, they are not printable.</p> <p>The <code>ungetch()</code> routine places <code>ch</code> back onto the input queue to be returned by the next call to <code>wgetch()</code>.</p> | | | | |
| Function Keys | <p>The following function keys, defined in <code><curses.h></code>, might be returned by <code>getch()</code> if <code>keypad()</code> has been enabled. Note that not all of these may be supported on a particular terminal if the terminal does not transmit a unique code when the key is pressed or if the definition for the key is not present in the <i>terminfo</i> database.</p> <table><thead><tr><th>Name</th><th>Key name</th></tr></thead><tbody><tr><td><code>KEY_BREAK</code></td><td>Break key</td></tr></tbody></table> | Name | Key name | <code>KEY_BREAK</code> | Break key |
| Name | Key name | | | | |
| <code>KEY_BREAK</code> | Break key | | | | |

| <i>Name</i> | <i>Key name</i> |
|-------------------|---|
| KEY_DOWN | The four arrow keys . . . |
| KEY_UP | |
| KEY_LEFT | |
| KEY_RIGHT | |
| KEY_HOME | Home key (upward+left arrow) |
| KEY_BACKSPACE | Backspace |
| KEY_F0 | Function keys; space for 64 keys is reserved. |
| KEY_F(<i>n</i>) | For $0 \leq n \leq 63$ |
| KEY_DL | Delete line |
| KEY_IL | Insert line |
| KEY_DC | Delete character |
| KEY_IC | Insert char or enter insert mode |
| KEY_EIC | Exit insert char mode |
| KEY_CLEAR | Clear screen |
| KEY_EOS | Clear to end of screen |
| KEY_EOL | Clear to end of line |
| KEY_SF | Scroll 1 line forward |
| KEY_SR | Scroll 1 line backward (reverse) |
| KEY_NPAGE | Next page |
| KEY_PPAGE | Previous page |
| KEY_STAB | Set tab |
| KEY_CTAB | Clear tab |
| KEY_CATAB | Clear all tabs |
| KEY_ENTER | Enter or send |
| KEY_SRESET | Soft (partial) reset |
| KEY_RESET | Reset or hard reset |
| KEY_PRINT | Print or copy |
| KEY_LL | Home down or bottom (lower left). Keypad is arranged like this: (Row 1) A1 up A3 (Row 2) left B2 right (Row 3) C1 down C3 |

curs_getch(3CURSES)

| <i>Name</i> | <i>Key name</i> |
|---------------|-----------------------|
| KEY_A1 | Upper left of keypad |
| KEY_A3 | Upper right of keypad |
| KEY_B2 | Center of keypad |
| KEY_C1 | Lower left of keypad |
| KEY_C3 | Lower right of keypad |
| KEY_BTAB | Back tab key |
| KEY_BEG | Beg(inning) key |
| KEY_CANCEL | Cancel key |
| KEY_CLOSE | Close key |
| KEY_COMMAND | Cmd (command) key |
| KEY_COPY | Copy key |
| KEY_CREATE | Create key |
| KEY_END | End key |
| KEY_EXIT | Exit key |
| KEY_FIND | Find key |
| KEY_HELP | Help key |
| KEY_MARK | Mark key |
| KEY_MESSAGE | Message key |
| KEY_MOVE | Move key |
| KEY_NEXT | Next object key |
| KEY_OPEN | Open key |
| KEY_OPTIONS | Options key |
| KEY_PREVIOUS | Previous object key |
| KEY_REDO | Redo key |
| KEY_REFERENCE | Reference key |
| KEY_REFRESH | Refresh key |
| KEY_REPLACE | Replace key |
| KEY_RESTART | Restart key |
| KEY_RESUME | Resume key |

| <i>Name</i> | <i>Key name</i> |
|---------------|-------------------------|
| KEY_SAVE | Save key |
| KEY_SBEG | Shifted beginning key |
| KEY_SCANCEL | Shifted cancel key |
| KEY_SCOMMAND | Shifted command key |
| KEY_SCOPY | Shifted copy key |
| KEY_SCREATE | Shifted create key |
| KEY_SDC | Shifted delete char key |
| KEY_SDL | Shifted delete line key |
| KEY_SELECT | Select key |
| KEY_SEND | Shifted end key |
| KEY_SEOL | Shifted clear line key |
| KEY_SEXIT | Shifted exit key |
| KEY_SFIND | Shifted find key |
| KEY_SHELP | Shifted help key |
| KEY_SHOME | Shifted home key |
| KEY_SIC | Shifted input key |
| KEY_SLEFT | Shifted left arrow key |
| KEY_SMESSAGE | Shifted message key |
| KEY_SMOVE | Shifted move key |
| KEY_SNEXT | Shifted next key |
| KEY_SOPTIONS | Shifted options key |
| KEY_SPREVIOUS | Shifted prev key |
| KEY_SPRINT | Shifted print key |
| KEY_SREDO | Shifted redo key |
| KEY_SREPLACE | Shifted replace key |
| KEY_SRIGHT | Shifted right arrow |
| KEY_SRSUME | Shifted resume key |
| KEY_SSAVE | Shifted save key |
| KEY_SSUSPEND | Shifted suspend key |

curs_getch(3CURSES)

| <i>Name</i> | <i>Key name</i> |
|-------------|------------------|
| KEY_SUNDO | Shifted undo key |
| KEY_SUSPEND | Suspend key |
| KEY_UNDO | Undo key |

RETURN VALUES All routines return the integer `ERR` upon failure. The `ungetch()` routine returns an integer value other than `ERR` upon successful completion. The other routines return the next input character or function key code upon successful completion.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|-----------------------|------------------------|
| MT-Level | Unsafe |

SEE ALSO `curs_inopts(3CURSES)`, `curs_move(3CURSES)`, `curs_refresh(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Use of the escape key for a single character function is discouraged.

When using `getch()`, `wgetch()`, `mvgetch()`, or `mvwgetch()`, `nocbreak` mode (`nocbreak()`) and `echo` mode (`echo()`) should not be used at the same time. Depending on the state of the tty driver when each character is typed, the program may produce undesirable results.

Note that `getch()`, `mvgetch()`, and `mvwgetch()` may be macros.

| NAME | curs_getstr, getstr, wgetstr, mvgetstr, mvwgetstr, wgetnstr – get character strings from curses terminal keyboard | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int getstr(char *str); int wgetstr(WINDOW *win, char *str); int mvgetstr(int y, int x, char *str); int mvwgetstr(WINDOW *win, int y, int x, char *str); int wgetnstr(WINDOW *win, char *str, int n);</pre> | | | | |
| DESCRIPTION | The effect of <code>getstr()</code> is as though a series of calls to <code>getch()</code> were made, until a newline or carriage return is received. The resulting value is placed in the area pointed to by the character pointer <code>str</code> . <code>wgetnstr()</code> reads at most <code>n</code> characters, thus preventing a possible overflow of the input buffer. The user's erase and kill characters are interpreted, as well as any special keys (such as function keys, HOME key, and CLEAR key.) | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_getch(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that <code>getstr()</code>, <code>mvgetstr()</code>, and <code>mvwgetstr()</code> may be macros.</p> | | | | |

curs_getwch(3CURSES)

| NAME | curs_getwch, getwch, wgetwch, mvwgetwch, mvwgetwch, ungetwch – get (or push back) wchar_t characters from curses terminal keyboard | | | | |
|----------------------|---|------|----------|-----------|-----------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int getwch(void); int wgetwch(WINDOW *win); int mvwgetwch(int y, int x); int mvwgetwch(WINDOW *win, int y, int x); int ungetwch(int wch);</pre> | | | | |
| DESCRIPTION | <p>The <code>getwch()</code>, <code>wgetwch()</code>, <code>mvwgetwch()</code>, and <code>mvwgetwch()</code> routines read an EUC character from the terminal associated with the window, transform it into a <code>wchar_t</code> character, and return a <code>wchar_t</code> character. In no-delay mode, if no input is waiting, the value <code>ERR</code> is returned. In delay mode, the program waits until the system passes text through to the program. Depending on the setting of <code>cbreak</code>, this is after one character (<code>cbreak</code> mode), or after the first newline (<code>nocbreak</code> mode). In half-delay mode, the program waits until a character is typed or the specified timeout has been reached. Unless <code>noecho</code> has been set, the character will also be echoed into the designated window.</p> <p>If the window is not a pad, and it has been moved or modified since the last call to <code>wrefresh(3CURSES)</code>, <code>wrefresh</code> will be called before another character is read.</p> <p>If <code>keypad</code> is <code>TRUE</code>, and a function key is pressed, the token for that function key is returned instead of the raw characters. Possible function keys are defined in <code><curses.h></code> with integers beginning with <code>0401</code>, whose names begin with <code>KEY_</code>. If a character that could be the beginning of a function key (such as escape) is received, <code>curses(3CURSES)</code> sets a timer. If the remainder of the sequence does not come in within the designated time, the character is passed through; otherwise, the function key value is returned. For this reason, many terminals experience a delay between the time a user presses the escape key and the escape is returned to the program.</p> <p>The <code>ungetwch()</code> routine places <code>wch</code> back onto the input queue to be returned by the next call to <code>wgetwch()</code>.</p> | | | | |
| Function Keys | <p>The following function keys, defined in <code><curses.h></code>, might be returned by <code>getwch()</code> if <code>keypad</code> has been enabled. Note that not all of these may be supported on a particular terminal if the terminal does not transmit a unique code when the key is pressed or if the definition for the key is not present in the <code>terminfo(4)</code> database.</p> <table border="1"><thead><tr><th>Name</th><th>Key name</th></tr></thead><tbody><tr><td>KEY_BREAK</td><td>Break key</td></tr></tbody></table> | Name | Key name | KEY_BREAK | Break key |
| Name | Key name | | | | |
| KEY_BREAK | Break key | | | | |

| <i>Name</i> | <i>Key name</i> |
|-------------------|---|
| KEY_DOWN | The four arrow keys . . . |
| KEY_UP | |
| KEY_LEFT | |
| KEY_RIGHT | |
| KEY_HOME | Home key (upward+left arrow) |
| KEY_BACKSPACE | Backspace |
| KEY_F0 | Function keys; space for 64 keys is reserved. |
| KEY_F(<i>n</i>) | For $0 \leq n \leq 63$ |
| KEY_DL | Delete line |
| KEY_IL | Insert line |
| KEY_DC | Delete character |
| KEY_IC | Insert char or enter insert mode |
| KEY_EIC | Exit insert char mode |
| KEY_CLEAR | Clear screen |
| KEY_EOS | Clear to end of screen |
| KEY_EOL | Clear to end of line |
| KEY_SF | Scroll 1 line forward |
| KEY_SR | Scroll 1 line backward (reverse) |
| KEY_NPAGE | Next page |
| KEY_PPAGE | Previous page |
| KEY_STAB | Set tab |
| KEY_CTAB | Clear tab |
| KEY_CATAB | Clear all tabs |
| KEY_ENTER | Enter or send |
| KEY_SRESET | Soft (partial) reset |
| KEY_RESET | Reset or hard reset |
| KEY_PRINT | Print or copy |
| KEY_LL | Home down or bottom (lower left). Keypad is arranged like this: A1 up A3 left B2 right C1 down C3 |

curs_getwch(3CURSES)

| <i>Name</i> | <i>Key name</i> |
|---------------|-----------------------|
| KEY_A1 | Upper left of keypad |
| KEY_A3 | Upper right of keypad |
| KEY_B2 | Center of keypad |
| KEY_C1 | Lower left of keypad |
| KEY_C3 | Lower right of keypad |
| KEY_BTAB | Back tab key |
| KEY_BEG | Beg(inning) key |
| KEY_CANCEL | Cancel key |
| KEY_CLOSE | Close key |
| KEY_COMMAND | Cmd (command) key |
| KEY_COPY | Copy key |
| KEY_CREATE | Create key |
| KEY_END | End key |
| KEY_EXIT | Exit key |
| KEY_FIND | Find key |
| KEY_HELP | Help key |
| KEY_MARK | Mark key |
| KEY_MESSAGE | Message key |
| KEY_MOVE | Move key |
| KEY_NEXT | Next object key |
| KEY_OPEN | Open key |
| KEY_OPTIONS | Options key |
| KEY_PREVIOUS | Previous object key |
| KEY_REDO | Redo key |
| KEY_REFERENCE | Reference key |
| KEY_REFRESH | Refresh key |
| KEY_REPLACE | Replace key |
| KEY_RESTART | Restart key |
| KEY_RESUME | Resume key |

| <i>Name</i> | <i>Key name</i> |
|---------------|-------------------------|
| KEY_SAVE | Save key |
| KEY_SBEG | Shifted beginning key |
| KEY_SCANCEL | Shifted cancel key |
| KEY_SCOMMAND | Shifted command key |
| KEY_SCOPY | Shifted copy key |
| KEY_SCREATE | Shifted create key |
| KEY_SDC | Shifted delete char key |
| KEY_SDL | Shifted delete line key |
| KEY_SELECT | Select key |
| KEY_SEND | Shifted end key |
| KEY_SEOL | Shifted clear line key |
| KEY_SEXIT | Shifted exit key |
| KEY_SFIND | Shifted find key |
| KEY_SHELP | Shifted help key |
| KEY_SHOME | Shifted home key |
| KEY_SIC | Shifted input key |
| KEY_SLEFT | Shifted left arrow key |
| KEY_SMESSAGE | Shifted message key |
| KEY_SMOVE | Shifted move key |
| KEY_SNEXT | Shifted next key |
| KEY_SOPTIONS | Shifted options key |
| KEY_SPREVIOUS | Shifted prev key |
| KEY_SPRINT | Shifted print key |
| KEY_SREDO | Shifted redo key |
| KEY_SREPLACE | Shifted replace key |
| KEY_SRIGHT | Shifted right arrow |
| KEY_SRSUME | Shifted resume key |
| KEY_SSAVE | Shifted save key |
| KEY_SSUSPEND | Shifted suspend key |

curs_getwch(3CURSES)

| <i>Name</i> | <i>Key name</i> |
|-------------|------------------|
| KEY_SUNDO | Shifted undo key |
| KEY_SUSPEND | Suspend key |
| KEY_UNDO | Undo key |

RETURN VALUE All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|-----------------------|------------------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), curs_inopts(3CURSES), curs_move(3CURSES), wrefresh(3CURSES), terminfo(4), attributes(5)

NOTES The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <widec.h>.

Use of the escape key by a programmer for a single character function is discouraged.

When using getch(), wgetch(), mvgetch(), or mvwgetch(), nocbreak mode and echo mode should not be used at the same time. Depending on the state of the tty driver when each character is typed, the program may produce undesirable results.

Note that getch(), mvgetch(), and mvwgetch() may be macros.

NAME | curs_getwstr, getwstr, getnwstr, wgetwstr, wgetnwstr, mvgetwstr, mvgetnwstr, mvwgetwstr, mvwgetnwstr – get wchar_t character strings from curses terminal keyboard

SYNOPSIS |

```
cc [ flag ... ] file ... -lcurses [ library .. ]
#include <curses.h>

int getwstr(wchar_t *wstr);
int getnwstr(wchar_t *wstr, int n);
int wgetwstr(WINDOW *win, wchar_t *wstr);
int wgetnwstr(WINDOW *win, wchar_t *wstr, int n);
int mvgetwstr(int y, int x, wchar_t *wstr);
int mvgetnwstr(int y, int x, wchar_t *wstr, int n);
int mvwgetwstr(WINDOW *win, int y, int x, wchar_t *wstr);
int mvwgetnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);
```

DESCRIPTION | The effect of getwstr() is as though a series of calls to getch(3CURSES) were made, until a newline and carriage return is received. The resulting value is placed in the area pointed to by the wchar_t pointer wstr. getnwstr() reads at most n wchar_t characters, thus preventing a possible overflow of the input buffer. The user's erase and kill characters are interpreted, as well as any special keys (such as function keys, HOME key, CLEAR key, etc.).

RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES | See attributes(5) for a description of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | curses(3CURSES), getch(3CURSES), attributes(5)

NOTES | The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h>, and <wctype.h>.

Note that all routines except wgetnwstr() may be macros.

curs_getyx(3CURSES)

| NAME | curs_getyx, getyx, getparyx, getbegyx, getmaxyx – get curses cursor and window coordinates | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> void getyx(WINDOW *win, int y, int x); void getparyx(WINDOW *win, int y, int x); void getbegyx(WINDOW *win, int y, int x); void getmaxyx(WINDOW *win, int y, int x);</pre> | | | | |
| DESCRIPTION | <p>With the <code>getyx()</code> macro, the cursor position of the window is placed in the two integer variables <code>y</code> and <code>x</code>.</p> <p>With the <code>getparyx()</code> macro, if <code>win</code> is a subwindow, the beginning coordinates of the subwindow relative to the parent window are placed into two integer variables, <code>y</code> and <code>x</code>. Otherwise, <code>-1</code> is placed into <code>y</code> and <code>x</code>.</p> <p>Like <code>getyx()</code>, the <code>getbegyx()</code> and <code>getmaxyx()</code> macros store the current beginning coordinates and size of the specified window.</p> | | | | |
| RETURN VALUES | The return values of these macros are undefined (that is, they should not be used as the right-hand side of assignment statements). | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that all of these interfaces are macros and that “&” is not necessary before the variables <code>y</code> and <code>x</code>.</p> | | | | |

NAME curs_inch, inch, winch, mvinch, mvwinch – get a character and its attributes from a curses window

SYNOPSIS

```
cc [ flag ... ] file ... -lcurses [ library ... ]
#include <curses.h>

chtype inch(void);
chtype winch(WINDOW *win);
chtype mvinch(int y, int x);
chtype mvwinch(WINDOW *win, int y, int x);
```

DESCRIPTION With these routines, the character, of type chtype, at the current position in the named window is returned. If any attributes are set for that position, their values are OR-ed into the value returned. Constants defined in <curses.h> can be used with the logical AND (&) operator to extract the character or attributes alone.

Attributes The following bit-masks can be AND-ed with characters returned by winch().

| | |
|--------------|--|
| A_CHARTEXT | Bit-mask to extract character |
| A_ATTRIBUTES | Bit-mask to extract attributes |
| A_COLOR | Bit-mask to extract color-pair field information |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), attributes(5)

NOTES The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.

Note that all of these routines may be macros.

curs_inchstr(3CURSES)

| NAME | <code>curs_inchstr</code> , <code>inchstr</code> , <code>inchnstr</code> , <code>winchstr</code> , <code>winchnstr</code> , <code>mvinchstr</code> , <code>mvinchnstr</code> , <code>mvwinchstr</code> , <code>mvwinchnstr</code> – get a string of characters (and attributes) from a curses window | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int inchstr(chtype *chstr); int inchnstr(chtype *chstr, int n); int winchstr(WINDOW *win, chtype *chstr); int winchnstr(WINDOW *win, chtype *chstr, int n); int mvinchstr(int y, int x, chtype *chstr); int mvinchnstr(int y, int x, chtype *chstr, int n); int mvwinchstr(WINDOW *win, int y, int x, chtype *chstr); int mvwinchnstr(WINDOW *win, int y, int x, chtype *chstr, int n);</pre> | | | | |
| DESCRIPTION | With these routines, a string of type <code>chtype</code> , starting at the current cursor position in the named window and ending at the right margin of the window, is returned. The four functions with <i>n</i> as the last argument, return the string at most <i>n</i> characters long. Constants defined in <code><curses.h></code> can be used with the <code>&</code> (logical AND) operator to extract the character or the attribute alone from any position in the <i>chstr</i> (see <code>curs_inch(3CURSES)</code>). | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_inch(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . Note that all routines except <code>winchnstr()</code> may be macros. | | | | |

| | |
|--------------------|--|
| NAME | curs_initscr, initscr, newterm, endwin, isendwin, set_term, delscreen – curses screen initialization and manipulation routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> WINDOW *initscr(void); int endwin(void); int isendwin(void); SCREEN *newterm(char *<i>type</i>, FILE *<i>outfd</i>, FILE *<i>infd</i>); SCREEN *set_term(SCREEN *<i>new</i>); void delscreen(SCREEN * <i>sp</i>);</pre> |
| DESCRIPTION | <p><code>initscr()</code> is almost always the first routine that should be called (the exceptions are <code>slk_init()</code>, <code>filter()</code>, <code>ripoffline()</code>, <code>use_env()</code> and, for multiple-terminal applications, <code>newterm()</code>.) This determines the terminal type and initializes all curses data structures. <code>initscr()</code> also causes the first call to <code>refresh()</code> to clear the screen. If errors occur, <code>initscr()</code> writes an appropriate error message to standard error and exits; otherwise, a pointer is returned to <code>stdscr()</code>. If the program needs an indication of error conditions, <code>newterm()</code> should be used instead of <code>initscr()</code>; <code>initscr()</code> should only be called once per application.</p> <p>A program that outputs to more than one terminal should use the <code>newterm()</code> routine for each terminal instead of <code>initscr()</code>. A program that needs an indication of error conditions, so it can continue to run in a line-oriented mode if the terminal cannot support a screen-oriented program, would also use this routine. The routine <code>newterm()</code> should be called once for each terminal. It returns a variable of type <code>SCREEN *</code> which should be saved as a reference to that terminal. The arguments are the <i>type</i> of the terminal to be used in place of <code>\$TERM</code>, a file pointer for output to the terminal, and another file pointer for input from the terminal (if <i>type</i> is <code>NULL</code>, <code>\$TERM</code> will be used). The program must also call <code>endwin()</code> for each terminal being used before exiting from curses. If <code>newterm()</code> is called more than once for the same terminal, the first terminal referred to must be the last one for which <code>endwin()</code> is called.</p> <p>A program should always call <code>endwin()</code> before exiting or escaping from curses mode temporarily. This routine restores tty modes, moves the cursor to the lower left-hand corner of the screen and resets the terminal into the proper non-visual mode. Calling <code>refresh()</code> or <code>doupdate()</code> after a temporary escape causes the program to resume visual mode.</p> <p>The <code>isendwin()</code> routine returns <code>TRUE</code> if <code>endwin()</code> has been called without any subsequent calls to <code>wrefresh()</code>, and <code>FALSE</code> otherwise.</p> |

`curs_initscr(3CURSES)`

The `set_term()` routine is used to switch between different terminals. The screen reference `new` becomes the new current terminal. The previous terminal is returned by the routine. This is the only routine which manipulates `SCREEN` pointers; all other routines affect only the current terminal.

The `delscreen()` routine frees storage associated with the `SCREEN` data structure. The `endwin()` routine does not do this, so `delscreen()` should be called after `endwin()` if a particular `SCREEN` is no longer needed.

RETURN VALUES `endwin()` returns the integer `ERR` upon failure and `OK` upon successful completion.

Routines that return pointers always return `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_kernel(3CURSES)`, `curs_refresh(3CURSES)`, `curs_slk(3CURSES)`, `curs_util(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `initscr()` and `newterm()` may be macros.

| | |
|--------------------|---|
| NAME | curs_inopts, cbreak, nocbreak, echo, noecho, halfdelay, intrflush, keypad, meta, nodelay, notimeout, raw, noraw, noqiflush, qiflush, timeout, wtimeout, typeahead – curses terminal input option control routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int cbreak(void); int nocbreak(void); int echo(void); int noecho(void); int halfdelay(int tenths); int intrflush(WINDOW *win, bool bf); int keypad(WINDOW *win, bool bf); int meta(WINDOW *win, bool bf); int nodelay(WINDOW *win, bool bf); int notimeout(WINDOW *win, bool bf); int raw(void); int noraw(void); void noqiflush(void); void qiflush(void); void timeout(int delay); void wtimeout(WINDOW *win, int delay); int typeahead(int fildes);</pre> |
| DESCRIPTION | <p>The <code>cbreak()</code> and <code>nocbreak()</code> routines put the terminal into and out of <code>cbreak()</code> mode, respectively. In this mode, characters typed by the user are immediately available to the program, and erase/kill character-processing is not performed. When out of this mode, the tty driver buffers the typed characters until a newline or carriage return is typed. Interrupt and flow control characters are unaffected by this mode. Initially the terminal may or may not be in <code>cbreak()</code> mode, as the mode is inherited; therefore, a program should call <code>cbreak()</code> or <code>nocbreak()</code> explicitly. Most interactive programs using <code>curses</code> set the <code>cbreak()</code> mode.</p> <p>Note that <code>cbreak()</code> overrides <code>raw()</code>. (See <code>curs_getch(3CURSES)</code> for a discussion of how these routines interact with <code>echo()</code> and <code>noecho()</code>.)</p> <p>The <code>echo()</code> and <code>noecho()</code> routines control whether characters typed by the user are echoed by <code>getch()</code> as they are typed. Echoing by the tty driver is always disabled, but initially <code>getch()</code> is in <code>echo</code> mode, so characters typed are echoed. Authors of</p> |

curs_inopts(3CURSES)

most interactive programs prefer to do their own echoing in a controlled area of the screen, or not to echo at all, so they disable echoing by calling `noecho()`. (See `curs_getch(3CURSES)` for a discussion of how these routines interact with `cbreak()` and `nocbreak()`.)

The `halfdelay()` routine is used for half-delay mode, which is similar to `cbreak()` mode in that characters typed by the user are immediately available to the program. However, after blocking for *tenths* tenths of seconds, `ERR` is returned if nothing has been typed. The value of *tenths* must be a number between 1 and 255. Use `nocbreak()` to leave half-delay mode.

If the `intrflush()` option is enabled, (*bf* is `TRUE`), when an interrupt key is pressed on the keyboard (interrupt, break, quit) all output in the tty driver queue will be flushed, giving the effect of faster response to the interrupt, but causing `curses` to have the wrong idea of what is on the screen. Disabling (*bf* is `FALSE`), the option prevents the flush. The default for the option is inherited from the tty driver settings. The window argument is ignored.

The `keypad()` option enables the keypad of the user's terminal. If enabled (*bf* is `TRUE`), the user can press a function key (such as an arrow key) and `wgetch()` returns a single value representing the function key, as in `KEY_LEFT`. If disabled (*bf* is `FALSE`), `curses` does not treat function keys specially and the program has to interpret the escape sequences itself. If the keypad in the terminal can be turned on (made to transmit) and off (made to work locally), turning on this option causes the terminal keypad to be turned on when `wgetch()` is called. The default value for keypad is false.

Initially, whether the terminal returns 7 or 8 significant bits on input depends on the control mode of the tty driver (see `termio(7I)`). To force 8 bits to be returned, invoke `meta(win, TRUE)`. To force 7 bits to be returned, invoke `meta(win, FALSE)`. The window argument, *win*, is always ignored. If the terminfo capabilities `smm` (`meta_on`) and `rmm` (`meta_off`) are defined for the terminal, `smm` is sent to the terminal when `meta(win, TRUE)` is called and `rmm` is sent when `meta(win, FALSE)` is called.

The `nodelay()` option causes `getch()` to be a non-blocking call. If no input is ready, `getch()` returns `ERR`. If disabled (*bf* is `FALSE`), `getch()` waits until a key is pressed.

While interpreting an input escape sequence, `wgetch()` sets a timer while waiting for the next character. If `notimeout(win, TRUE)` is called, then `wgetch()` does not set a timer. The purpose of the timeout is to differentiate between sequences received from a function key and those typed by a user.

With the `raw()` and `noraw()` routines, the terminal is placed into or out of raw mode. Raw mode is similar to `cbreak()` mode, in that characters typed are immediately passed through to the user program. The differences are that in raw mode, the interrupt, quit, suspend, and flow control characters are all passed through uninterpreted, instead of generating a signal. The behavior of the `BREAK` key depends on other bits in the tty driver that are not set by `curses`.

`curs_inopts(3CURSES)`

When the `noqiflush()` routine is used, normal flush of input and output queues associated with the `INTR`, `QUIT` and `SUSP` characters will not be done (see `termio(7I)`). When `qiflush()` is called, the queues will be flushed when these control characters are read.

The `timeout()` and `wtimeout()` routines set blocking or non-blocking read for a given window. If *delay* is negative, blocking read is used (that is, waits indefinitely for input). If *delay* is zero, then non-blocking read is used (that is, read returns `ERR` if no input is waiting). If *delay* is positive, then read blocks for *delay* milliseconds, and returns `ERR` if there is still no input. Hence, these routines provide the same functionality as `nodelay()`, plus the additional capability of being able to block for only *delay* milliseconds (where *delay* is positive).

`curses` does “line-breakout optimization” by looking for typeahead periodically while updating the screen. If input is found, and it is coming from a `tty`, the current update is postponed until `refresh()` or `doupdate()` is called again. This allows faster response to commands typed in advance. Normally, the input `FILE` pointer passed to `newterm()`, or `stdin` in the case that `initscr()` was used, will be used to do this typeahead checking. The `typeahead()` routine specifies that the file descriptor *fildev* is to be used to check for typeahead instead. If *fildev* is `-1`, then no typeahead checking is done.

RETURN VALUES

All routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_getch(3CURSES)`, `curs_initscr(3CURSES)`, `curses(3CURSES)`, `attributes(5)`, `termio(7I)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `echo()`, `noecho()`, `halfdelay()`, `intrflush()`, `meta()`, `nodelay()`, `notimeout()`, `noqiflush()`, `qiflush()`, `timeout()`, and `wtimeout()` may be macros.

curs_insch(3CURSES)

NAME curs_insch, insch, winsch, mvwinsch, mvwinsch – insert a character before the character under the cursor in a curses window

SYNOPSIS

```
cc [ flag ... ] file... -lcurses [ library ... ]
#include <curses.h>

int insch(chtype ch);
int winsch(WINDOW *win, chtype ch);
int mvwinsch(int y, int x, chtype ch);
int mvwinsch(WINDOW *win, int y, int x, chtype ch);
```

DESCRIPTION With these routines, the character *ch* is inserted before the character under the cursor. All characters to the right of the cursor are moved one space to the right, with the possibility of the rightmost character on the line being lost. The cursor position does not change (after moving to *y*, *x*, if specified). (This does not imply use of the hardware insert character feature.)

RETURN VALUES All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), attributes(5)

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `insch()`, `mvwinsch()`, and `mvwinsch()` may be macros.

NAME curs_insstr, insstr, insnstr, winsstr, winsnstr, mvinsstr, mvinsnstr, mvwinsstr, mvwinsnstr – insert string before character under the cursor in a curses window

SYNOPSIS

```
cc [ flag ... ] file ... -lcurses [ library ... ]
#include <curses.h>

int insstr(char *str);
int insnstr(char *str, int n);
int winsstr(WINDOW *win, char *str);
int winsnstr(WINDOW *win, char *str, int n);
int mvinsstr(int y, int x, char *str);
int mvinsnstr(int y, int x, char *str, int n);
int mvwinsstr(WINDOW *win, int y, int x, char *str);
int mvwinsnstr(WINDOW *win, int y, int x, char *str, int n);
```

DESCRIPTION

With these routines, a character string (as many characters as will fit on the line) is inserted before the character under the cursor. All characters to the right of the cursor are moved to the right, with the possibility of the rightmost characters on the line being lost. The cursor position does not change (after moving to *y, x*, if specified). (This does not imply use of the hardware insert character feature.) The four routines with *n* as the last argument insert at most *n* characters. If *n* ≤ 0, then the entire string is inserted.

If a character in *str* is a tab, newline, carriage return or backspace, the cursor is moved appropriately within the window. A newline also does a `clrtoeol()` before moving. Tabs are considered to be at every eighth column. If a character in *str* is another control character, it is drawn in the `^X` notation. Calling `winch()` after adding a control character (and moving to it, if necessary) does not return the control character, but instead returns the representation of the control character.

RETURN VALUES All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_clear(3CURSES)`, `curs_inch(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

`curs_ insstr(3CURSES)`

Note that all but `winsnstr()` may be macros.

NAME curs_instr, instr, innstr, winstr, winnstr, mvinstr, mvinnstr, mvwinstr, mvwinnstr – get a string of characters from a curses window

SYNOPSIS `cc [flag ...] file ... -lcurses [library ...]
#include <curses.h>`

```
int instr(char *str);
int innstr(char *str, int n);
int winstr(WINDOW *win, char *str);
int winnstr(WINDOW *win, char *str, int n);
int mvinstr(int y, int x, char *str);
int mvinnstr(int y, int x, char *str, int n);
int mvwinstr(WINDOW *win, int y, int x, char *str);
int mvwinnstr(WINDOW *win, int y, int x, char *str, int n);
```

DESCRIPTION These routines return a string of characters in *str*, starting at the current cursor position in the named window and ending at the right margin of the window. Attributes are stripped from the characters. The four functions with *n* as the last argument return the string at most *n* characters long.

RETURN VALUES All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), attributes(5)

NOTES The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.

Note that all routines except winnstr() may be macros.

curs_inswch(3CURSES)

| NAME | <code>curs_inswch</code> , <code>inswch</code> , <code>winswch</code> , <code>mvinswch</code> , <code>mvwinswch</code> – insert a <code>wchar_t</code> character before the character under the cursor in a curses window | | | | |
|---------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ..] #include <curses.h> int inswch(<i>chtype wch</i>); int winswch(WINDOW *<i>win</i>, <i>chtype wch</i>); int mvinswch(int <i>y</i>, int <i>x</i>, <i>chtype wch</i>); int mvwinswch(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>, <i>chtype wch</i>);</pre> | | | | |
| DESCRIPTION | These routines insert the character <i>wch</i> , holding a <code>wchar_t</code> character, before the character under the cursor. All characters to the right of the cursor are moved one space to the right, with the possibility of the rightmost character on the line being lost. The cursor position does not change (after moving to <i>y</i> , <i>x</i> , if specified). (This does not imply use of the hardware insert character feature.) | | | | |
| RETURN VALUE | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header file <code><curses.h></code> automatically includes the header files <code><stdio.h></code> , <code><unctrl.h></code> and <code><wider.h></code> . Note that <code>inswch()</code> , <code>mvinswch()</code> , and <code>mvwinswch()</code> may be macros. None of these routines can use the color attribute in <code>chtype</code> . | | | | |

NAME curs_inswstr, inswstr, insnwstr, winswstr, winsnwstr, mvinswstr, mvinsnwstr, mvwinswstr, mvwinsnwstr – insert wchar_t string before character under the cursor in a curses window

SYNOPSIS

```
cc [ flag ... ] file ... -lcurses [ library .. ]
#include <curses.h>

int inswstr(wchar_t *wstr);
int insnwstr(wchar_t *wstr, int n);
int winswstr(WINDOW *win, wchar_t *wstr);
int winsnwstr(WINDOW *win, wchar_t *wstr, int n);
int mvinswstr(int y, int x, wchar_t *wstr);
int mvinsnwstr(int y, int x, wchar_t *wstr, int n);
int mvwinswstr(WINDOW *win, int y, int x, wchar_t *wstr);
int mvwinsnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);
```

DESCRIPTION These routines insert a wchar_t character string (as many wchar_t characters as will fit on the line) before the character under the cursor. All characters to the right of the cursor are moved to the right, with the possibility of the rightmost characters on the line being lost. The cursor position does not change (after moving to y, x, if specified). (This does not imply use of the hardware insert character feature.) The four routines with n as the last argument insert at most n wchar_t characters. If n<=0, then the entire string is inserted.

If a character in wstr is a tab, newline, carriage return, or backspace, the cursor is moved appropriately within the window. A newline also does a clrtoeol(3CURSES) before moving. Tabs are considered to be at every eighth column. If a character in wstr is another control character, it is drawn in the ^X notation. Calling winwch(3CURSES) after adding a control character (and moving to it, if necessary) does not return the control character, but instead returns the representation of the control character.

RETURN VALUE All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO clrtoeol(3CURSES), curses(3CURSES), winwch(3CURSES), attributes(5)

NOTES The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <wchar.h>.

`curs_inswstr(3CURSES)`

Note that all but `winsnwstr()` may be macros.

NAME curs_inwch, inwch, winwch, mvwinch, mvwinwch – get a wchar_t character and its attributes from a curses window

SYNOPSIS `cc [flag ...] file ... -lcurses [library ..]`
`#include <curses.h>`

`chtype inwch(void);`
`chtype winwch(WINDOW *win);`
`chtype mvwinch(int y, int x);`
`chtype mvwinwch(WINDOW *win, int y, int x);`

DESCRIPTION These routines return the wchar_t character, of type chtype, at the current position in the named window. If any attributes are set for that position, their values are OR-ed into the value returned. Constants defined in <curses.h> can be used with the logical AND (&) operator to extract the character or attributes alone.

Attributes The following bit-masks may be AND-ed with characters returned by winwch().

A_WCHARTEXT Bit-mask to extract character
A_WATTRIBUTES Bit-mask to extract attributes

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), attributes(5)

NOTES The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <wdec.h>.

Note that all of these routines may be macros.

None of these routines can use the color attribute in chtype.

curs_inwchstr(3CURSES)

| NAME | curs_inwchstr, inwchstr, inwchnstr, winwchstr, winwchnstr, mvinwchstr, mvinwchnstr, mvwinwchstr, mvwinwchnstr – get a string of wchar_t characters (and attributes) from a curses window | | | | |
|---------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int inwchstr (chtype *wchstr); int inwchnstr (chtype *wchstr, int n); int winwchstr (WINDOW *win, chtype *wchstr); int winwchnstr (WINDOW *win, chtype *wchstr, int n); int mvinwchstr (int y, int x, chtype *wchstr); int mvinwchnstr (int y, int x, chtype *wchstr, int n); int mvwinwchstr (WINDOW *win, int y, int x, chtype *wchstr); int mvwinwchnstr (WINDOW *win, int y, int x, chtype *wchstr, int n);</pre> | | | | |
| DESCRIPTION | <p>These routines return a string of type chtype, holding wchar_t characters, starting at the current cursor position in the named window and ending at the right margin of the window. The four functions with <i>n</i> as the last argument, return the string at most <i>n</i> wchar_t characters long. Constants defined in <curses.h> can be used with the logical AND (&) operator to extract the wchar_t character or the attribute alone from any position in the wchstr (see curs_inwch(3CURSES)).</p> | | | | |
| RETURN VALUE | <p>All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.</p> | | | | |
| ATTRIBUTES | <p>See attributes(5) for a description of the following attributes:</p> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">MT-Level</td> <td style="text-align: center;">Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <p>curses(3CURSES), curs_inwch(3CURSES), attributes(5)</p> | | | | |
| NOTES | <p>The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <wdec.h>.</p> <p>Note that all routines except winwchnstr() may be macros.</p> <p>None of these routines can use the color attribute in chtype.</p> | | | | |

NAME | curs_inwstr, inwstr, innwstr, winwstr, winnwstr, mvinwstr, mvinnwstr, mvwinwstr, mvwinnwstr – get a string of wchar_t characters from a curses window

SYNOPSIS |

```
cc [ flag ... ] file ... -lcurses[library .. ]
#include <curses.h>

int inwstr(wchar_t *wstr);
int innwstr(wchar_t *wstr, int n);
int winwstr(WINDOW *win, wchar_t *wstr);
int winnwstr(WINDOW *win, wchar_t *wstr, int n);
int mvinwstr(int y, int x, wchar_t *wstr);
int mvinnwstr(int y, int x, wchar_t *wstr, int n);
int mvwinwstr(WINDOW *win, int y, int x, wchar_t *wstr);
int mvwinnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);
```

DESCRIPTION | These routines return the string of wchar_t characters in *wstr* starting at the current cursor position in the named window and ending at the right margin of the window. Attributes are stripped from the characters. The four functions with *n* as the last argument return the string at most *n* wchar_t characters long.

RETURN VALUES | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | curses(3CURSES), attributes(5)

NOTES | The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <wider.h>.

Note that all routines except winnwstr() may be macros.

curs_kernel(3CURSES)

| | |
|--------------------|---|
| NAME | <code>curs_kernel</code> , <code>def_prog_mode</code> , <code>def_shell_mode</code> , <code>reset_prog_mode</code> , <code>reset_shell_mode</code> , <code>resetty</code> , <code>savetty</code> , <code>getsyx</code> , <code>setsyx</code> , <code>riporffline</code> , <code> curs_set</code> , <code>napms</code> – low-level curses routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int def_prog_mode(void); int def_shell_mode(void); int reset_prog_mode(void); int reset_shell_mode(void); int resetty(void); int savetty(void); int getsyx(int y, int x); int setsyx(int y, int x); int ripoffline(int line, int (*init)(WINDOW *, int)); int curs_set(int visibility); int napms(int ms);</pre> |
| DESCRIPTION | <p>The following routines give low-level access to various curses functionality. These routines typically are used inside library routines.</p> <p>The <code>def_prog_mode()</code> and <code>def_shell_mode()</code> routines save the current terminal modes as the “program” (in curses) or “shell” (not in curses) state for use by the <code>reset_prog_mode()</code> and <code>reset_shell_mode()</code> routines. This is done automatically by <code>initscr()</code>.</p> <p>The <code>reset_prog_mode()</code> and <code>reset_shell_mode()</code> routines restore the terminal to “program” (in curses) or “shell” (out of curses) state. These are done automatically by <code>endwin()</code> and, after an <code>endwin()</code>, by <code>doupdate()</code>, so they normally are not called.</p> <p>The <code>resetty()</code> and <code>savetty()</code> routines save and restore the state of the terminal modes. <code>savetty()</code> saves the current state in a buffer and <code>resetty()</code> restores the state to what it was at the last call to <code>savetty()</code>.</p> <p>With the <code>getsyx()</code> routine, the current coordinates of the virtual screen cursor are returned in <code>y</code> and <code>x</code>. If <code>leaveok()</code> is currently TRUE, then <code>-1,-1</code> is returned. If lines have been removed from the top of the screen, using <code>riporffline()</code>, <code>y</code> and <code>x</code> include these lines; therefore, <code>y</code> and <code>x</code> should be used only as arguments for <code>setsyx()</code>.</p> <p>With the <code>setsyx()</code> routine, the virtual screen cursor is set to <code>y, x</code>. If <code>y</code> and <code>x</code> are both <code>-1</code>, then <code>leaveok()</code> is set. The two routines <code>getsyx()</code> and <code>setsyx()</code> are designed to be used by a library routine, which manipulates curses windows but does not</p> |

want to change the current position of the program's cursor. The library routine would call `getsyx()` at the beginning, do its manipulation of its own windows, do a `wnoutrefresh()` on its windows, call `setsyx()`, and then call `doupdate()`.

The `riporffline()` routine provides access to the same facility that `slk_init()` (see `curs_slk(3CURSES)`) uses to reduce the size of the screen. `riporffline()` must be called before `initscr()` or `newterm()` is called. If `line` is positive, a line is removed from the top of `stdscr()`; if `line` is negative, a line is removed from the bottom. When this is done inside `initscr()`, the routine `init()` (supplied by the user) is called with two arguments: a window pointer to the one-line window that has been allocated and an integer with the number of columns in the window. Inside this initialization routine, the integer variables `LINES` and `COLS` (defined in `<curses.h>`) are not guaranteed to be accurate and `wrefresh()` or `doupdate()` must not be called. It is allowable to call `wnoutrefresh()` during the initialization routine.

`riporffline()` can be called up to five times before calling `initscr()` or `newterm()`.

With the `curs_set()` routine, the cursor state is set to invisible, normal, or very visible for *visibility* equal to 0, 1, or 2 respectively. If the terminal supports the *visibility* requested, the previous *cursor* state is returned; otherwise, `ERR` is returned.

The `napms()` routine is used to sleep for *ms* milliseconds.

RETURN VALUES

Except for `curs_set()`, these routines always return `OK`. `curs_set()` returns the previous cursor state, or `ERR` if the requested *visibility* is not supported.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_initscr(3CURSES)`, `curs_outopts(3CURSES)`, `curs_refresh(3CURSES)`, `curs_scr_dump(3CURSES)`, `curs_slk(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `getsyx()` is a macro, so an ampersand (&) is not necessary before the variables *y* and *x*.

curs_move(3CURSES)

NAME | curs_move, move, wmove – move curses window cursor

SYNOPSIS |

```
cc [ flag ... ] file ... -lcurses [ library ... ]
#include <curses.h>

int move(int y, int x);
int wmove(WINDOW *win, int y, int x);
```

DESCRIPTION | With these routines, the cursor associated with the window is moved to line *y* and column *x*. This routine does not move the physical cursor of the terminal until `refresh()` is called. The position specified is relative to the upper left-hand corner of the window, which is (0,0).

RETURN VALUES | These routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | `curs_refresh(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES | The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `move()` may be a macro.

| | |
|--------------------|---|
| NAME | curs_outopts, clearok, idlok, idcok, immedok, leaveok, setscreg, wsetscreg, scrollok, nl, nonl – curses terminal output option control routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int clearok(WINDOW *win, bool bf); int idlok(WINDOW *win, bool bf); void idcok(WINDOW *win, bool bf); void immedok(WINDOW *win, bool bf); int leaveok(WINDOW *win, bool bf); int setscreg(int top, int bot); int wsetscreg(WINDOW *win, int top, int bot); int scrollok(WINDOW *win, bool bf); int nl(void); int nonl(void);</pre> |
| DESCRIPTION | <p>These routines set options that deal with output within curses. All options are initially FALSE, unless otherwise stated. It is not necessary to turn these options off before calling <code>endwin()</code>.</p> <p>With the <code>clearok()</code> routine, if enabled (<i>bf</i> is TRUE), the next call to <code>wrefresh()</code> with this window will clear the screen completely and redraw the entire screen from scratch. This is useful when the contents of the screen are uncertain, or in some cases for a more pleasing visual effect. If the <i>win</i> argument to <code>clearok()</code> is the global variable <code>curscr()</code>, the next call to <code>wrefresh()</code> with any window causes the screen to be cleared and repainted from scratch.</p> <p>With the <code>idlok()</code> routine, if enabled (<i>bf</i> is TRUE), <code>curses</code> considers using the hardware insert/delete line feature of terminals so equipped. If disabled (<i>bf</i> is FALSE), <code>curses</code> very seldom uses this feature. (The insert/delete character feature is always considered.) This option should be enabled only if the application needs insert/delete line, for example, for a screen editor. It is disabled by default because insert/delete line tends to be visually annoying when used in applications where it isn't really needed. If insert/delete line cannot be used, <code>curses</code> redraws the changed portions of all lines.</p> <p>With the <code>idcok()</code> routine, if enabled (<i>bf</i> is TRUE), <code>curses</code> considers using the hardware insert/delete character feature of terminals so equipped. This is enabled by default.</p> <p>With the <code>immedok()</code> routine, if enabled (<i>bf</i> is TRUE), any change in the window image, such as the ones caused by <code>waddch()</code>, <code>wclrtoobot()</code>, <code>wscrl()</code>, etc., automatically cause a call to <code>wrefresh()</code>. However, it may degrade the performance considerably, due to repeated calls to <code>wrefresh()</code>. It is disabled by default.</p> |

curs_outopts(3CURSES)

Normally, the hardware cursor is left at the location of the window cursor being refreshed. The `leaveok()` option allows the cursor to be left wherever the update happens to leave it. It is useful for applications where the cursor is not used, since it reduces the need for cursor motions. If possible, the cursor is made invisible when this option is enabled.

The `setscrreg()` and `wsetscrreg()` routines allow the application programmer to set a software scrolling region in a window. *top* and *bot* are the line numbers of the top and bottom margin of the scrolling region. (Line 0 is the top line of the window.) If this option and `scrollok()` are enabled, an attempt to move off the bottom margin line causes all lines in the scrolling region to scroll up one line. Only the text of the window is scrolled. (Note that this has nothing to do with the use of a physical scrolling region capability in the terminal, like that in the VT100. If `idlok()` is enabled and the terminal has either a scrolling region or insert/delete line capability, they will probably be used by the output routines.)

The `scrollok()` option controls what happens when the cursor of a window is moved off the edge of the window or scrolling region, either as a result of a newline action on the bottom line, or typing the last character of the last line. If disabled, (*bf* is FALSE), the cursor is left on the bottom line. If enabled, (*bf* is TRUE), `wrefresh()` is called on the window, and the physical terminal and window are scrolled up one line. (Note that in order to get the physical scrolling effect on the terminal, it is also necessary to call `idlok()`.)

The `nl()` and `nonl()` routines control whether newline is translated into carriage return and linefeed on output, and whether return is translated into newline on input. Initially, the translations do occur. By disabling these translations using `nonl()`, `curses` is able to make better use of the linefeed capability, resulting in faster cursor motion.

RETURN VALUES `setscrreg()` and `wsetscrreg()` return OK upon success and ERR upon failure. All other routines that return an integer always return OK.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_addch(3CURSES)`, `curs_clear(3CURSES)`, `curs_initscr(3CURSES)`, `curs_refresh(3CURSES)`, `curs_scroll(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `clearok()`, `leaveok()`, `scrollok()`, `idcok()`, `nl()`, `nonl()`, and `setscrreg()` may be macros.

`curs_outopts(3CURSES)`

The `immedok()` routine is useful for windows that are used as terminal emulators.

curs_overlay(3CURSES)

| NAME | curs_overlay, overlay, overwrite, copywin – overlap and manipulate overlapped curses windows | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int overlay(WINDOW *srcwin, WINDOW *dstwin); int overwrite(WINDOW *srcwin, WINDOW *dstwin); int copywin(WINDOW *srcwin, WINDOW *dstwin, int sminrow, int smincol, int dminrow, int dmincol, int dmaxrow, int dmaxcol, int overlay);</pre> | | | | |
| DESCRIPTION | <p>The <code>overlay()</code> and <code>overwrite()</code> routines overlay <code>srcwin</code> on top of <code>dstwin</code>. <code>srcwin</code> and <code>dstwin</code> are not required to be the same size; only text where the two windows overlap is copied. The difference is that <code>overlay()</code> is non-destructive (blanks are not copied) whereas <code>overwrite()</code> is destructive.</p> <p>The <code>copywin()</code> routine provides a finer granularity of control over the <code>overlay()</code> and <code>overwrite()</code> routines. Like in the <code>prefresh()</code> routine, a rectangle is specified in the destination window, (<code>dminrow</code>, <code>dmincol</code>) and (<code>dmaxrow</code>, <code>dmaxcol</code>), and the upper-left-corner coordinates of the source window, (<code>sminrow</code>, <code>smincol</code>). If the argument <code>overlay</code> is true, then copying is non-destructive, as in <code>overlay()</code>.</p> | | | | |
| RETURN VALUES | Routines that return an integer return ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_pad(3CURSES)</code> , <code>curs_refresh(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that <code>overlay()</code> and <code>overwrite</code> may be macros.</p> | | | | |

| | |
|--------------------|--|
| NAME | curs_pad, newpad, subpad, prefresh, pnoutrefresh, pechochar, pechowchar – create and display curses pads |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> WINDOW *newpad(int nlines, int ncols); WINDOW *subpad(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); int prefresh(WINDOW *pad, int pminrow, int pmincol, int sminrow, int smincol, int smaxrow, int smaxcol); int pnoutrefresh(WINDOW *pad, int pminrow, int pmincol, int sminrow, int smincol, int smaxrow, int smaxcol); int pechochar(WINDOW *pad, chtype ch); int pechowchar(WINDOW *pad, chtype wch);</pre> |
| DESCRIPTION | <p>The <code>newpad()</code> routine creates and returns a pointer to a new pad data structure with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. A pad is like a window, except that it is not restricted by the screen size, and is not necessarily associated with a particular part of the screen. Pads can be used when a large window is needed, and only a part of the window will be on the screen at one time. Automatic refreshes of pads (for example, from scrolling or echoing of input) do not occur. It is not legal to call <code>wrefresh(3CURSES)</code> with a <i>pad</i> as an argument; the routines <code>prefresh()</code> or <code>pnoutrefresh()</code> should be called instead. Note that these routines require additional parameters to specify the part of the pad to be displayed and the location on the screen to be used for the display.</p> <p>The <code>subpad()</code> routine creates and returns a pointer to a subwindow within a pad with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. Unlike <code>subwin(3CURSES)</code>, which uses screen coordinates, the window is at position (<i>begin_x</i>, <i>begin_y</i>) on the pad. The window is made in the middle of the window <i>orig</i>, so that changes made to one window affect both windows. During the use of this routine, it will often be necessary to call <code>touchwin(3CURSES)</code> or <code>touchline(3CURSES)</code> on <i>orig</i> before calling <code>prefresh()</code>.</p> <p>The <code>prefresh()</code> and <code>pnoutrefresh()</code> routines are analogous to <code>wrefresh(3CURSES)</code> and <code>wnoutrefresh(3CURSES)</code> except that they relate to pads instead of windows. The additional parameters are needed to indicate what part of the pad and screen are involved. <i>pminrow</i> and <i>pmincol</i> specify the upper left-hand corner of the rectangle to be displayed in the pad. <i>sminrow</i>, <i>smincol</i>, <i>smaxrow</i>, and <i>smaxcol</i> specify the edges of the rectangle to be displayed on the screen. The lower right-hand corner of the rectangle to be displayed in the pad is calculated from the screen coordinates, since the rectangles must be the same size. Both rectangles must be entirely contained within their respective structures. Negative values of <i>pminrow</i>, <i>pmincol</i>, <i>sminrow</i>, or <i>smincol</i> are treated as if they were zero.</p> |

`curs_pad(3CURSES)`

The `pechochar()` routine is functionally equivalent to a call to `addch(3CURSES)` followed by a call to `refresh(3CURSES)`, a call to `waddch(3CURSES)` followed by a call to `wrefresh(3CURSES)`, or a call to `waddch(3CURSES)` followed by a call to `prefresh()`. The knowledge that only a single character is being output is taken into consideration and, for non-control characters, a considerable performance gain might be seen by using these routines instead of their equivalents. In the case of `pechochar()`, the last location of the pad on the screen is reused for the arguments to `prefresh()`.

RETURN VALUES Routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

Routines that return pointers return `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `addch(3CURSES)`, `curses(3CURSES)`, `refresh(3CURSES)`, `subwin(3CURSES)`, `touchline(3CURSES)`, `touchwin(3CURSES)`, `waddch(3CURSES)`, `wnoutrefresh(3CURSES)`, `wrefresh(3CURSES)`, `attributes(5)`

NOTES The header file `<curses.h>` automatically includes the header files `<stdio.h>`, `<unctrl.h>` and `<widec.h>`.

Note that `pechochar()` may be a macro.

NAME curs_printw, printw, wprintw, mvprintw, mvwprintw, vwprintw – print formatted output in curses windows

SYNOPSIS

```
cc [ flag ... ] file ... -lcurses [ library ... ]
#include <curses.h>

int printw(char *fmt, /* arg */ ... );
int wprintw(WINDOW *win, char *fmt, /* arg */ ... );
int mvprintw(int y, int x, char *fmt, /* arg */ ... );
int mvwprintw(WINDOW *win, int y, int x, char *fmt, /* arg */... );
#include <varargs.h>
int vwprintw(WINDOW *win, char *fmt, /* varlist */ ... );
```

DESCRIPTION

The `printw()`, `wprintw()`, `mvprintw()`, and `mvwprintw()` routines are analogous to `printf()` (see `printf(3C)`). In effect, the string that would be output by `printf()` is output instead as though `waddstr()` were used on the given window.

The `vwprintw()` routine is analogous to `vprintf()` (see `vprintf(3C)`) and performs a `wprintw()` using a variable argument list. The third argument is a `va_list`, a pointer to a list of arguments, as defined in `<varargs.h>`.

RETURN VALUES All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3CURSES)`, `printf(3C)`, `vprintf(3C)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

curs_refresh(3CURSES)

| | |
|--------------------|---|
| NAME | <code>curs_refresh</code> , <code>refresh</code> , <code>wrefresh</code> , <code>wnoutrefresh</code> , <code>doupdate</code> , <code>redrawwin</code> , <code>wredrawln</code> – refresh curses windows and lines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int refresh(void); int wrefresh(WINDOW *<i>win</i>); int wnoutrefresh(WINDOW *<i>win</i>); int doupdate(void); int redrawwin(WINDOW *<i>win</i>); int wredrawln(WINDOW *<i>win</i>, int <i>beg_line</i>, int <i>num_lines</i>);</pre> |
| DESCRIPTION | <p>The <code>refresh()</code> and <code>wrefresh()</code> routines (or <code>wnoutrefresh()</code> and <code>doupdate()</code>) must be called to get any output on the terminal, as other routines merely manipulate data structures. The routine <code>wrefresh()</code> copies the named window to the physical terminal screen, taking into account what is already there in order to do optimizations. The <code>refresh()</code> routine is the same, using <code>stdscr</code> as the default window. Unless <code>leaveok()</code> has been enabled, the physical cursor of the terminal is left at the location of the cursor for that window.</p> <p>The <code>wnoutrefresh()</code> and <code>doupdate()</code> routines allow multiple updates with more efficiency than <code>wrefresh()</code> alone. In addition to all the window structures, <code>curses</code> keeps two data structures representing the terminal screen: a physical screen, describing what is actually on the screen, and a virtual screen, describing what the programmer wants to have on the screen.</p> <p>The routine <code>wrefresh()</code> works by first calling <code>wnoutrefresh()</code>, which copies the named window to the virtual screen, and then calling <code>doupdate()</code>, which compares the virtual screen to the physical screen and does the actual update. If the programmer wishes to output several windows at once, a series of calls to <code>wrefresh()</code> results in alternating calls to <code>wnoutrefresh()</code> and <code>doupdate()</code>, causing several bursts of output to the screen. By first calling <code>wnoutrefresh()</code> for each window, it is then possible to call <code>doupdate()</code> once, resulting in only one burst of output, with fewer total characters transmitted and less CPU time used. If the <code>win</code> argument to <code>wrefresh()</code> is the global variable <code>curscr</code>, the screen is immediately cleared and repainted from scratch.</p> <p>The <code>redrawwin()</code> routine indicates to <code>curses</code> that some screen lines are corrupted and should be thrown away before anything is written over them. These routines could be used for programs such as editors, which want a command to redraw some part of the screen or the entire screen. The routine <code>wredrawln()</code> is preferred over <code>redrawwin()</code> where a noisy communication line exists and redrawing the entire window could be subject to even more communication noise. Just redrawing several lines offers the possibility that they would show up unblemished.</p> |

`curs_refresh(3CURSES)`

RETURN VALUES All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_outopts(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `refresh()` and `redrawwin()` may be macros.

curs_scanw(3CURSES)

| NAME | curs_scanw, scanw, wscanw, mvscanw, mvwscanw, vwscanw – convert formatted input from a curses widow | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int scanw(char *fmt, /* arg */ ...); int wscanw(WINDOW *win, char *fmt, /* arg */ ...); int mvscanw(int y, int x, char *fmt, /* arg */ ...); int mvwscanw(WINDOW *win, int y, int x, char *fmt, /* arg */...); int vwscanw(WINDOW *win, char *fmt, va_list varglist);</pre> | | | | |
| DESCRIPTION | <p>The scanw(), wscanw(), and mvscanw() routines correspond to scanf() (see scanf(3C)). The effect of these routines is as though wgetstr() were called on the window, and the resulting line used as input for the scan. Fields which do not map to a variable in the fmt field are lost.</p> <p>The vwscanw() routine is similar to vwprintw() in that it performs a wscanw() using a variable argument list. The third argument is a va_list, a pointer to a list of arguments, as defined in <varargs.h>.</p> | | | | |
| RETURN VALUES | <p>wscanw() returns ERR on failure and an integer equal to the number of fields scanned on success.</p> <p>Applications may interrogate the return value from the scanw, wscanw(), mvscanw(), and mvwscanw() routines to determine the number of fields which were mapped in the call.</p> | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curs_getstr(3CURSES), curs_printw(3CURSES), curses(3CURSES), scanf(3C), attributes(5) | | | | |
| NOTES | The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>. | | | | |

NAME curs_scr_dump, scr_dump, scr_restore, scr_init, scr_set – read (write) a curses screen from (to) a file

SYNOPSIS

```
cc [ flag ... ] file ... -lcurses [ library ... ]
#include <curses.h>

int scr_dump(char *filename);
int scr_restore(char *filename);
int scr_init(char *filename);
int scr_set(char *filename);
```

DESCRIPTION

With the `scr_dump()` routine, the current contents of the virtual screen are written to the file *filename*.

With the `scr_restore()` routine, the virtual screen is set to the contents of *filename*, which must have been written using `scr_dump()`. The next call to `doupdate()` restores the screen to the way it looked in the dump file.

With the `scr_init()` routine, the contents of *filename* are read in and used to initialize the curses data structures about what the terminal currently has on its screen. If the data is determined to be valid, curses bases its next update of the screen on this information rather than clearing the screen and starting from scratch. `scr_init()` is used after `initscr()` or a `system(3C)` call to share the screen with another process which has done a `scr_dump()` after its `endwin()` call. The data is declared invalid if the time-stamp of the tty is old or the terminfo capabilities `rmcup()` and `nrrmc()` exist.

The `scr_set()` routine is a combination of `scr_restore()` and `scr_init()`. It tells the program that the information in *filename* is what is currently on the screen, and also what the program wants on the screen. This can be thought of as a screen inheritance function.

To read (write) a window from (to) a file, use the `getwin()` and `putwin()` routines (see `curs_util(3CURSES)`).

RETURN VALUES All routines return the integer ERR upon failure and OK upon success.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_initscr(3CURSES)`, `curs_refresh(3CURSES)`, `curs_util(3CURSES)`, `curses(3CURSES)`, `system(3C)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

`curs_scr_dump(3CURSES)`

Note that `scr_init()`, `scr_set()`, and `scr_restore()` may be macros.

NAME | curs_scroll, scroll, scl, wscrl – scroll a curses window

SYNOPSIS |

```
cc [ flag ... ] file ... -lcurses [ library ... ]
#include <curses.h>

int scroll(WINDOW *win);
int scl(int n);
int wscrl(WINDOW *win, int n);
```

DESCRIPTION | With the scroll() routine, the window is scrolled up one line. This involves moving the lines in the window data structure. As an optimization, if the scrolling region of the window is the entire screen, the physical screen is scrolled at the same time.

With the scl() and wscrl() routines, for positive *n* scroll the window up *n* lines (line *i+n* becomes *i*); otherwise scroll the window down *n* lines. This involves moving the lines in the window character image structure. The current cursor position is not changed.

For these functions to work, scrolling must be enabled via scrollok().

RETURN VALUES | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | curs_outopts(3CURSES), curses(3CURSES), attributes(5)

NOTES | The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.

Note that scl() and scroll() may be macros.

curs_set(3CURSES)

| | |
|--------------------|--|
| NAME | curs_kernel, def_prog_mode, def_shell_mode, reset_prog_mode, reset_shell_mode, resetty, savetty, getsyx, setsyx, ripoffline, curs_set, napms – low-level curses routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int def_prog_mode(void); int def_shell_mode(void); int reset_prog_mode(void); int reset_shell_mode(void); int resetty(void); int savetty(void); int getsyx(int y, int x); int setsyx(int y, int x); int ripoffline(int line, int (*init)(WINDOW *, int)); int curs_set(int visibility); int napms(int ms);</pre> |
| DESCRIPTION | <p>The following routines give low-level access to various curses functionality. These routines typically are used inside library routines.</p> <p>The <code>def_prog_mode()</code> and <code>def_shell_mode()</code> routines save the current terminal modes as the “program” (in curses) or “shell” (not in curses) state for use by the <code>reset_prog_mode()</code> and <code>reset_shell_mode()</code> routines. This is done automatically by <code>initscr()</code>.</p> <p>The <code>reset_prog_mode()</code> and <code>reset_shell_mode()</code> routines restore the terminal to “program” (in curses) or “shell” (out of curses) state. These are done automatically by <code>endwin()</code> and, after an <code>endwin()</code>, by <code>doupdate()</code>, so they normally are not called.</p> <p>The <code>resetty()</code> and <code>savetty()</code> routines save and restore the state of the terminal modes. <code>savetty()</code> saves the current state in a buffer and <code>resetty()</code> restores the state to what it was at the last call to <code>savetty()</code>.</p> <p>With the <code>getsyx()</code> routine, the current coordinates of the virtual screen cursor are returned in <code>y</code> and <code>x</code>. If <code>leaveok()</code> is currently TRUE, then <code>-1,-1</code> is returned. If lines have been removed from the top of the screen, using <code>ripoffline()</code>, <code>y</code> and <code>x</code> include these lines; therefore, <code>y</code> and <code>x</code> should be used only as arguments for <code>setsyx()</code>.</p> <p>With the <code>setsyx()</code> routine, the virtual screen cursor is set to <code>y, x</code>. If <code>y</code> and <code>x</code> are both <code>-1</code>, then <code>leaveok()</code> is set. The two routines <code>getsyx()</code> and <code>setsyx()</code> are designed to be used by a library routine, which manipulates curses windows but does not</p> |

want to change the current position of the program's cursor. The library routine would call `getsyx()` at the beginning, do its manipulation of its own windows, do a `wnoutrefresh()` on its windows, call `setsyx()`, and then call `doupdate()`.

The `riporffline()` routine provides access to the same facility that `slk_init()` (see `curs_slk(3CURSES)`) uses to reduce the size of the screen. `riporffline()` must be called before `initscr()` or `newterm()` is called. If `line` is positive, a line is removed from the top of `stdscr()`; if `line` is negative, a line is removed from the bottom. When this is done inside `initscr()`, the routine `init()` (supplied by the user) is called with two arguments: a window pointer to the one-line window that has been allocated and an integer with the number of columns in the window. Inside this initialization routine, the integer variables `LINES` and `COLS` (defined in `<curses.h>`) are not guaranteed to be accurate and `wrefresh()` or `doupdate()` must not be called. It is allowable to call `wnoutrefresh()` during the initialization routine.

`riporffline()` can be called up to five times before calling `initscr()` or `newterm()`.

With the `curs_set()` routine, the cursor state is set to invisible, normal, or very visible for *visibility* equal to 0, 1, or 2 respectively. If the terminal supports the *visibility* requested, the previous *cursor* state is returned; otherwise, `ERR` is returned.

The `napms()` routine is used to sleep for *ms* milliseconds.

RETURN VALUES

Except for `curs_set()`, these routines always return `OK`. `curs_set()` returns the previous cursor state, or `ERR` if the requested *visibility* is not supported.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_initscr(3CURSES)`, `curs_outopts(3CURSES)`, `curs_refresh(3CURSES)`, `curs_scr_dump(3CURSES)`, `curs_slk(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `getsyx()` is a macro, so an ampersand (&) is not necessary before the variables *y* and *x*.

curs_set(3XCURSES)

NAME | curs_set – set visibility of cursor

SYNOPSIS |

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

int curs_set(int visibility);
```

DESCRIPTION | The curs_set() function sets the visibility of the cursor to invisible (0), normal (1), or very visible (2). The exact appearance of normal and very visible cursors is terminal dependent.

PARAMETERS | *visibility* Is a value of 0 (invisible), 1 (normal), or 2 (very visible).

RETURN VALUES | If the terminal supports the mode specified by the *visibility* parameter, the curs_set() function returns the previous cursor state. Otherwise, it returns ERR.

ERRORS | None.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | libcurses(3XCURSES), attributes(5), standards(5)

| | |
|--------------------|--|
| NAME | curs_slk, slk_init, slk_set, slk_refresh, slk_noutrefresh, slk_label, slk_clear, slk_restore, slk_touch, slk_atron, slk_attrset, slk_attroff – curses soft label routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int slk_init(int <i>fmt</i>) ; int slk_set(int <i>labnum</i>, char *<i>label</i>, int <i>fmt</i>) ; int slk_refresh(void) ; int slk_noutrefresh(void) ; char *slk_label(int <i>labnum</i>) ; int slk_clear(void) ; int slk_restore(void) ; int slk_touch(void) ; int slk_atron(chtype <i>attrs</i>) ; int slk_attrset(chtype <i>attrs</i>) ; int slk_attroff(chtype <i>attrs</i>) ;</pre> |
| DESCRIPTION | <p>curses manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, <i>curses</i> takes over the bottom line of <code>stdscr</code>, reducing the size of <code>stdscr</code> and the variable <code>LINES</code>. <i>curses</i> standardizes on eight labels of up to eight characters each.</p> <p>To use soft labels, the <code>slk_init()</code> routine must be called before <code>initscr()</code> or <code>newterm()</code> is called. If <code>initscr()</code> eventually uses a line from <code>stdscr</code> to emulate the soft labels, then <code>fmt</code> determines how the labels are arranged on the screen. Setting <code>fmt</code> to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement.</p> <p>With the <code>slk_set()</code> routine, <i>labnum</i> is the label number, from 1 to 8. <i>label</i> is the string to be put on the label, up to eight characters in length. A null string or a null pointer sets up a blank label. <code>fmt</code> is either 0, 1, or 2, indicating whether the label is to be left-justified, centered, or right-justified, respectively, within the label.</p> <p>The <code>slk_refresh()</code> and <code>slk_noutrefresh()</code> routines correspond to the <code>wrefresh()</code> and <code>wnoutrefresh()</code> routines.</p> <p>With the <code>slk_label()</code> routine, the current label for label number <i>labnum</i> is returned with leading and trailing blanks stripped.</p> <p>With the <code>slk_clear()</code> routine, the soft labels are cleared from the screen.</p> <p>With the <code>slk_restore()</code> routine, the soft labels are restored to the screen after a <code>slk_clear()</code> is performed.</p> |

`curl_slk(3CURSES)`

With the `slk_touch()` routine, all the soft labels are forced to be output the next time a `slk_noutrefresh()` is performed.

The `slk_attron()`, `slk_attrset()`, and `slk_attroff()` routines correspond to `attron()`, `attrset()`, and `attroff()`. They have an effect only if soft labels are simulated on the bottom line of the screen.

RETURN VALUES Routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`slk_label()` returns `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_attr(3CURSES)`, `curl_initscr(3CURSES)`, `curl_refresh(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.

| | |
|----------------------|--|
| NAME | curs_termattrs, baudrate, erasechar, has_ic, has_il, killchar, longname, termattrs, termname – curses environment query routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int baudrate(void); char erasechar(void); int has_ic(void); int has_il(void); char killchar(void); char *longname(void); cctype termattrs(void); char *termname(void);</pre> |
| DESCRIPTION | <p>The <code>baudrate()</code> routine returns the output speed of the terminal. The number returned is in bits per second, for example 9600, and is an integer.</p> <p>With the <code>erasechar()</code> routine, the user's current erase character is returned.</p> <p>The <code>has_ic()</code> routine is true if the terminal has insert- and delete-character capabilities.</p> <p>The <code>has_il()</code> routine is true if the terminal has insert- and delete-line capabilities, or can simulate them using scrolling regions. This might be used to determine if it would be appropriate to turn on physical scrolling using <code>scrolllok()</code>.</p> <p>With the <code>killchar()</code> routine, the user's current line kill character is returned.</p> <p>The <code>longname()</code> routine returns a pointer to a static area containing a verbose description of the current terminal. The maximum length of a verbose description is 128 characters. It is defined only after the call to <code>initscr()</code> or <code>newterm()</code>. The area is overwritten by each call to <code>newterm()</code> and is not restored by <code>set_term()</code>, so the value should be saved between calls to <code>newterm()</code> if <code>longname()</code> is going to be used with multiple terminals.</p> <p>If a given terminal doesn't support a video attribute that an application program is trying to use, <code>curses</code> may substitute a different video attribute for it. The <code>termattrs()</code> function returns a logical OR of all video attributes supported by the terminal. This information is useful when a <code>curses</code> program needs complete control over the appearance of the screen.</p> <p>The <code>termname()</code> routine returns the value of the environment variable <code>TERM</code> (truncated to 14 characters).</p> |
| RETURN VALUES | <code>longname()</code> and <code>termname()</code> return <code>NULL</code> on error. |

`curs_termattrs(3CURSES)`

Routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_initscr(3CURSES)`, `curs_outopts(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `termattrs()` may be a macro.

NAME | curs_termcap, tgetent, tgetflag, tgetnum, tgetstr, tgoto, tputs – curses interfaces (emulated) to the termcap library

SYNOPSIS |

```
cc [ flag ... ] file ... -lcurses [ library ... ]
#include <curses.h>
#include <term.h>

int tgetent(char *bp, char *name);
int tgetflag(char id[2]);
int tgetnum(char id[2]);
char *tgetstr(char id[2], char **area);
char *tgoto(char *cap, int col, int row);
int tputs(char *str, int affcnt, int (*putc)(void));
```

DESCRIPTION | These routines are included as a conversion aid for programs that use the *termcap* library. Their parameters are the same and the routines are emulated using the *terminfo* database. These routines are supported at Level 2 and should not be used in new applications.

The `tgetent()` routine looks up the termcap entry for *name*. The emulation ignores the buffer pointer *bp*.

The `tgetflag()` routine gets the boolean entry for *id*.

The `tgetnum()` routine gets the numeric entry for *id*.

The `tgetstr()` routine returns the string entry for *id*. Use `tputs()` to output the returned string.

The `tgoto()` routine instantiates the parameters into the given capability. The output from this routine is to be passed to `tputs()`.

The `tputs()` routine is described on the `curs_terminfo(3CURSES)` manual page.

RETURN VALUES | Routines that return an integer return ERR upon failure and an integer value other than ERR upon successful completion.

Routines that return pointers return NULL on error.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | `curs_terminfo(3CURSES)`, `curses(3CURSES)`, `putc(3C)`, `attributes(5)`

`curl_terminap(3CURSES)`

NOTES | The header `<curls.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

| | |
|--------------------|--|
| NAME | curs_terminfo, setupterm, setterm, set_curterm, del_curterm, restartterm, tparm, tputs, putp, vidputs, vidattr, mvcur, tigetflag, tigetnum, tigetstr – curses interfaces to terminfo database |
| SYNOPSIS | <pre> cc [flag ...] file ... -lcurses [library ...] #include <curses.h> #include <term.h> int setupterm(char *term, int fildes, int *errret); int setterm(char *term); int set_curterm(TERMINAL *nterm); int del_curterm(TERMINAL *oterm); int restartterm(char *term, int fildes, int *errret); char *tparm(char *str, long int p1, long int p2, long int p3, long int p4, long int p5, long int p6, long int p7, long int p8, long int p9); int tputs(char *str, int affcnt, int (*putc)(char)); int putp(char *str); int vidputs(chtype attrs, int (*putc)(char)); int vidattr(chtype attrs); int mvcur(int oldrow, int oldcol, int newrow, int newcol); int tigetflag(char *capname); int tigetnum(char *capname); char *tigetstr(char *capname); </pre> |
| DESCRIPTION | <p>These low-level routines must be called by programs that have to deal directly with the <i>terminfo</i> database to handle certain terminal capabilities, such as programming function keys. For all other functionality, curses routines are more suitable and their use is recommended.</p> <p>Initially, <code>setupterm()</code> should be called. Note that <code>setupterm()</code> is automatically called by <code>initscr()</code> and <code>newterm()</code>. This defines the set of terminal-dependent variables (listed in <code>terminfo(4)</code>). The <i>terminfo</i> variables <code>lines</code> and <code>columns</code> are initialized by <code>setupterm()</code> as follows: If <code>use_env(FALSE)</code> has been called, values for <code>lines</code> and <code>columns</code> specified in <i>terminfo</i> are used. Otherwise, if the environment variables <code>LINES</code> and <code>COLUMNS</code> exist, their values are used. If these environment variables do not exist and the program is running in a window, the current window size is used. Otherwise, if the environment variables do not exist, the values for <code>lines</code> and <code>columns</code> specified in the <i>terminfo</i> database are used.</p> <p>The headers <code><curses.h></code> and <code><term.h></code> should be included (in this order) to get the definitions for these strings, numbers, and flags. Parameterized strings should be passed through <code>tparm()</code> to instantiate them. All <i>terminfo</i> strings (including the output</p> |

curs_terminfo(3CURSES)

of `tparm()` should be printed with `tputs()` or `putp()`. Call the `reset_shell_mode()` routine to restore the tty modes before exiting (see `curs_kernel(3CURSES)`). Programs which use cursor addressing should output `enter_ca_mode` upon startup and should output `exit_ca_mode` before exiting. Programs desiring shell escapes should call `reset_shell_mode` and output `exit_ca_mode` before the shell is called and should output `enter_ca_mode` and call `reset_prog_mode` after returning from the shell.

The `setupterm()` routine reads in the *terminfo* database, initializing the *terminfo* structures, but does not set up the output virtualization structures used by *curses*. The terminal type is the character string *term*; if *term* is null, the environment variable `TERM` is used. All output is to file descriptor *fildev* which is initialized for output. If *errret* is not null, then `setupterm()` returns `OK` or `ERR` and stores a status value in the integer pointed to by *errret*. A status of 1 in *errret* is normal, 0 means that the terminal could not be found, and -1 means that the *terminfo* database could not be found. If *errret* is null, `setupterm()` prints an error message upon finding an error and exits. Thus, the simplest call is:

```
setupterm((char *)0, 1, (int *)0);
```

which uses all the defaults and sends the output to `stdout`.

The `setterm()` routine is being replaced by `setupterm()`. The call:

```
setupterm(term, 1, (int *)0)
```

provides the same functionality as `setterm(term)`. The `setterm()` routine is included here for compatibility and is supported at Level 2.

The `set_curterm()` routine sets the variable `cur_term` to *nterm*, and makes all of the *terminfo* boolean, numeric, and string variables use the values from *nterm*.

The `del_curterm()` routine frees the space pointed to by *oterm* and makes it available for further use. If *oterm* is the same as `cur_term`, references to any of the *terminfo* boolean, numeric, and string variables thereafter may refer to invalid memory locations until another `setupterm()` has been called.

The `restartterm()` routine is similar to `setupterm()` and `initscr()`, except that it is called after restoring memory to a previous state. It assumes that the windows and the input and output options are the same as when memory was saved, but the terminal type and baud rate may be different.

The `tparm()` routine instantiates the string *str* with parameters *pi*. A pointer is returned to the result of *str* with the parameters applied.

The `tputs()` routine applies padding information to the string *str* and outputs it. The *str* must be a *terminfo* string variable or the return value from `tparm()`, `tgetstr()`, or `tgoto()`. *affcnt* is the number of lines affected, or 1 if not applicable. *putc* is a `putchar()`-like routine to which the characters are passed, one at a time.

`curs_terminfo(3CURSES)`

The `putp()` routine calls `tputs(str, 1, putchar)`. Note that the output of `putpA()` always goes to `stdout`, not to the *files* specified in `setupterm()`.

The `vidputs()` routine displays the string on the terminal in the video attribute mode *attrs*, which is any combination of the attributes listed in `curses(3CURSES)`. The characters are passed to the `putchar()`-like routine `putc()`.

The `vidattr()` routine is like the `vidputs()` routine, except that it outputs through `putchar()`.

The `mvcur()` routine provides low-level cursor motion.

The `tigetflag()`, `tigetnum()` and `tigetstr()` routines return the value of the capability corresponding to the *terminfo capname* passed to them, such as `xenl`.

With the `tigetflag()` routine, the value `-1` is returned if *capname* is not a boolean capability.

With the `tigetnum()` routine, the value `-2` is returned if *capname* is not a numeric capability.

With the `tigetstr()` routine, the value `(char *)-1` is returned if *capname* is not a string capability.

The *capname* for each capability is given in the table column entitled *capname* code in the capabilities section of `terminfo(4)`.

```
char *boolnames, *boolcodes, *boolfnames
char *numnames, *numcodes, *numfnames
char *strnames, *strcodes, *strfnames
```

These null-terminated arrays contain the *capnames*, the *termcap* codes, and the full C names, for each of the *terminfo* variables.

RETURN VALUES

All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

Routines that return pointers always return `NULL` on error.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_initscr(3CURSES)`, `curs_kernel(3CURSES)`, `curs_termcap(3CURSES)`, `curses(3CURSES)`, `putc(3C)`, `terminfo(4)`, `attributes(5)`

`curs_terminfo(3CURSES)`

NOTES | The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

The `setupterm()` routine should be used in place of `setterm()`.

Note that `vidattr()` and `vidputs()` may be macros.

NAME curs_touch, touchwin, touchline, untouchwin, wtouchln, is_linetouched, is_wintouched – curses refresh control routines

SYNOPSIS

```
cc [ flag ... ] file ... -lcurses [ library ... ]
#include <curses.h>

int touchwin(WINDOW *win);
int touchline(WINDOW *win, int start, int count);
int untouchwin(WINDOW *win);
int wtouchln(WINDOW *win, int y, int n, int changed);
int is_linetouched(WINDOW *win, int line);
int is_wintouched(WINDOW *win);
```

DESCRIPTION

The touchwin() and touchline() routines throw away all optimization information about which parts of the window have been touched, by pretending that the entire window has been drawn on. This is sometimes necessary when using overlapping windows, since a change to one window affects the other window, but the records of which lines have been changed in the other window do not reflect the change. The routine touchline() only pretends that *count* lines have been changed, beginning with line *start*.

The untouchwin() routine marks all lines in the window as unchanged since the last call to wrefresh().

The wtouchln() routine makes *n* lines in the window, starting at line *y*, look as if they have (*changed*=1) or have not (*changed*=0) been changed since the last call to wrefresh().

The is_linetouched() and is_wintouched() routines return TRUE if the specified line/window was modified since the last call to wrefresh(); otherwise they return FALSE. In addition, is_linetouched() returns ERR if line is not valid for the given window.

RETURN VALUES All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion, unless otherwise noted in the preceding routine descriptions.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_refresh(3CURSES), curses(3CURSES), attributes(5)

NOTES The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.

`curs_touch(3CURSES)`

Note that all routines except `wtouchln()` may be macros.

| | |
|--------------------|--|
| NAME | curs_util, unctrl, keyname, filter, use_env, putwin, getwin, delay_output, flushinp – curses miscellaneous utility routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> char *unctrl (chtype <i>c</i>); char *keyname (int <i>c</i>); int filter (void); void use_env (char <i>bool</i>); int putwin (WINDOW *<i>win</i>, FILE *<i>filep</i>); WINDOW *getwin (FILE *<i>filep</i>); int delay_output (int <i>ms</i>); int flushinp (void);</pre> |
| DESCRIPTION | <p>The unctrl () macro expands to a character string which is a printable representation of the character <i>c</i>. Control characters are displayed in the ^X notation. Printing characters are displayed as is.</p> <p>With the keyname () routine, a character string corresponding to the key <i>c</i> is returned.</p> <p>The filter () routine, if used, is called before initscr () or newterm () are called. It makes curses think that there is a one-line screen. curses does not use any terminal capabilities that assume that they know on what line of the screen the cursor is positioned.</p> <p>The use_env () routine, if used, is called before initscr () or newterm () are called. When called with FALSE as an argument, the values of lines and columns specified in the terminfo database will be used, even if environment variables LINES and COLUMNS (used by default) are set, or if curses is running in a window (in which case default behavior would be to use the window size if LINES and COLUMNS are not set).</p> <p>With the putwin () routine, all data associated with window <i>win</i> is written into the file to which <i>filep</i> points. This information can be later retrieved using the getwin () function.</p> <p>The getwin () routine reads window related data stored in the file by putwin (). The routine then creates and initializes a new window using that data. It returns a pointer to the new window.</p> <p>The delay_output () routine inserts an <i>ms</i> millisecond pause in output. This routine should not be used extensively because padding characters are used rather than a CPU pause.</p> <p>The flushinp () routine throws away any typeahead that has been typed by the user and has not yet been read by the program.</p> |

`curs_util(3CURSES)`

RETURN VALUES Except for `flushinp()`, routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`flushinp()` always returns `OK`.

Routines that return pointers return `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_initscr(3CURSES)`, `curs_scr_dump(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `unctrl()` is a macro, which is defined in `<unctrl.h>`.

| | |
|--------------------|--|
| NAME | curs_window, newwin, delwin, mvwin, subwin, derwin, mvderwin, dupwin, wsyncup, syncok, wcursyncup, wsyncdown – create curses windows |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> WINDOW *newwin(int nlines, int ncols, int begin_y, int begin_x); int delwin(WINDOW *win); int mvwin(WINDOW *win, int y, int x); WINDOW *subwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); WINDOW *derwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); int mvderwin(WINDOW *win, int par_y, int par_x); WINDOW *dupwin(WINDOW *win); void wsyncup(WINDOW *win); int syncok(WINDOW *win, bool bf); void wcursyncup(WINDOW *win); void wsyncdown(WINDOW *win);</pre> |
| DESCRIPTION | <p>The <code>newwin()</code> routine creates and returns a pointer to a new window with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. The upper left-hand corner of the window is at line <i>begin_y</i>, column <i>begin_x</i>. If either <i>nlines</i> or <i>ncols</i> is zero, they default to LINES — <i>begin_y</i> and COLS — <i>begin_x</i>. A new full-screen window is created by calling <code>newwin(0, 0, 0, 0)</code>.</p> <p>The <code>delwin()</code> routine deletes the named window, freeing all memory associated with it. Subwindows must be deleted before the main window can be deleted.</p> <p>The <code>mvwin()</code> routine moves the window so that the upper left-hand corner is at position (<i>x</i>, <i>y</i>). If the move would cause the window to be off the screen, it is an error and the window is not moved. Moving subwindows is allowed, but should be avoided.</p> <p>The <code>subwin()</code> routine creates and returns a pointer to a new window with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. The window is at position (<i>begin_y</i>, <i>begin_x</i>) on the screen. (This position is relative to the screen, and not to the window <i>orig</i>.) The window is made in the middle of the window <i>orig</i>, so that changes made to one window will affect both windows. The subwindow shares memory with the window <i>orig</i>. When using this routine, it is necessary to call <code>touchwin()</code> or <code>touchline()</code> on <i>orig</i> before calling <code>wrefresh()</code> on the subwindow.</p> <p>The <code>derwin()</code> routine is the same as <code>subwin()</code>, except that <i>begin_y</i> and <i>begin_x</i> are relative to the origin of the window <i>orig</i> rather than the screen. There is no difference between the subwindows and the derived windows.</p> |

curs_window(3CURSES)

The `mvderwin()` routine moves a derived window (or subwindow) inside its parent window. The screen-relative parameters of the window are not changed. This routine is used to display different parts of the parent window at the same physical position on the screen.

The `dupwin()` routine creates an exact duplicate of the window *win*.

Each `curses` window maintains two data structures: the character image structure and the status structure. The character image structure is shared among all windows in the window hierarchy (that is, the window with all subwindows). The status structure, which contains information about individual line changes in the window, is private to each window. The routine `wrefresh()` uses the status data structure when performing screen updating. Since status structures are not shared, changes made to one window in the hierarchy may not be properly reflected on the screen.

The routine `wsyncup()` causes the changes in the status structure of a window to be reflected in the status structures of its ancestors. If `syncok()` is called with second argument `TRUE` then `wsyncup()` is called automatically whenever there is a change in the window.

The routine `wcursyncup()` updates the current cursor position of all the ancestors of the window to reflect the current cursor position of the window.

The routine `wsyncdown()` updates the status structure of the window to reflect the changes in the status structures of its ancestors. Applications seldom call this routine because it is called automatically by `wrefresh()`.

RETURN VALUES

Routines that return an integer return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`delwin()` returns the integer `ERR` upon failure and `OK` upon successful completion.

Routines that return pointers return `NULL` on error.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_refresh(3CURSES)`, `curs_touch(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

If many small changes are made to the window, the `wsyncup()` option could degrade performance.

`curs_window(3CURSES)`

Note that `syncok()` may be a macro.

cur_term(3XCURSES)

- NAME** cur_term – current terminal information
- SYNOPSIS**

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

extern TERMINAL *cur_term;
```
- DESCRIPTION** The external variable `cur_term` to identifies the record in the `terminfo` associated with the terminal currently in use.
- ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

- SEE ALSO** `libcurses(3XCURSES)`, `set_curterm(3XCURSES)`, `tigetflag(3XCURSES)`, `attributes(5)`, `standards(5)`

data_ahead(3CURSES)

NAME form_data, data_ahead, data_behind – tell if forms field has off-screen data ahead or behind

SYNOPSIS

```
cc [ flag ... ] file ... -lform -lcurses [ library .. ]
#include <form.h>

int data_ahead(FORM *form) ;
int data_behind(FORM *form) ;
```

DESCRIPTION data_ahead() returns TRUE (1) if the current field has more off-screen data ahead; otherwise it returns FALSE (0).
data_behind() returns TRUE (1) if the current field has more off-screen data behind; otherwise it returns FALSE (0).

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), forms(3CURSES), attributes(5)

NOTES The header <form.h> automatically includes the headers <eti.h> and <curses.h>.

data_behind(3CURSES)

| NAME | form_data, data_ahead, data_behind – tell if forms field has off-screen data ahead or behind | | | | |
|--------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lform -lcurses [library ..] #include <form.h> int data_ahead(FORM *form) ; int data_behind(FORM *form) ;</pre> | | | | |
| DESCRIPTION | <p>data_ahead() returns TRUE (1) if the current field has more off-screen data ahead; otherwise it returns FALSE (0).</p> <p>data_behind() returns TRUE (1) if the current field has more off-screen data behind; otherwise it returns FALSE (0).</p> | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | |

| | |
|--------------------|--|
| NAME | curs_kernel, def_prog_mode, def_shell_mode, reset_prog_mode, reset_shell_mode, resetty, savetty, getsyx, setsyx, ripoffline, curs_set, napms – low-level curses routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int def_prog_mode(void); int def_shell_mode(void); int reset_prog_mode(void); int reset_shell_mode(void); int resetty(void); int savetty(void); int getsyx(int y, int x); int setsyx(int y, int x); int ripoffline(int line, int (*init)(WINDOW *, int)); int curs_set(int visibility); int napms(int ms);</pre> |
| DESCRIPTION | <p>The following routines give low-level access to various curses functionality. These routines typically are used inside library routines.</p> <p>The <code>def_prog_mode()</code> and <code>def_shell_mode()</code> routines save the current terminal modes as the “program” (in curses) or “shell” (not in curses) state for use by the <code>reset_prog_mode()</code> and <code>reset_shell_mode()</code> routines. This is done automatically by <code>initscr()</code>.</p> <p>The <code>reset_prog_mode()</code> and <code>reset_shell_mode()</code> routines restore the terminal to “program” (in curses) or “shell” (out of curses) state. These are done automatically by <code>endwin()</code> and, after an <code>endwin()</code>, by <code>doupdate()</code>, so they normally are not called.</p> <p>The <code>resetty()</code> and <code>savetty()</code> routines save and restore the state of the terminal modes. <code>savetty()</code> saves the current state in a buffer and <code>resetty()</code> restores the state to what it was at the last call to <code>savetty()</code>.</p> <p>With the <code>getsyx()</code> routine, the current coordinates of the virtual screen cursor are returned in <code>y</code> and <code>x</code>. If <code>leaveok()</code> is currently TRUE, then <code>-1,-1</code> is returned. If lines have been removed from the top of the screen, using <code>ripoffline()</code>, <code>y</code> and <code>x</code> include these lines; therefore, <code>y</code> and <code>x</code> should be used only as arguments for <code>setsyx()</code>.</p> <p>With the <code>setsyx()</code> routine, the virtual screen cursor is set to <code>y, x</code>. If <code>y</code> and <code>x</code> are both <code>-1</code>, then <code>leaveok()</code> is set. The two routines <code>getsyx()</code> and <code>setsyx()</code> are designed to be used by a library routine, which manipulates curses windows but does not</p> |

def_prog_mode(3CURSES)

want to change the current position of the program's cursor. The library routine would call `getsyx()` at the beginning, do its manipulation of its own windows, do a `wnoutrefresh()` on its windows, call `setsyx()`, and then call `doupdate()`.

The `riporffline()` routine provides access to the same facility that `slk_init()` (see `curs_slk(3CURSES)`) uses to reduce the size of the screen. `riporffline()` must be called before `initscr()` or `newterm()` is called. If `line` is positive, a line is removed from the top of `stdscr()`; if `line` is negative, a line is removed from the bottom. When this is done inside `initscr()`, the routine `init()` (supplied by the user) is called with two arguments: a window pointer to the one-line window that has been allocated and an integer with the number of columns in the window. Inside this initialization routine, the integer variables `LINES` and `COLS` (defined in `<curses.h>`) are not guaranteed to be accurate and `wrefresh()` or `doupdate()` must not be called. It is allowable to call `wnoutrefresh()` during the initialization routine.

`riporffline()` can be called up to five times before calling `initscr()` or `newterm()`.

With the `curs_set()` routine, the cursor state is set to invisible, normal, or very visible for *visibility* equal to 0, 1, or 2 respectively. If the terminal supports the *visibility* requested, the previous *cursor* state is returned; otherwise, `ERR` is returned.

The `napms()` routine is used to sleep for *ms* milliseconds.

RETURN VALUES Except for `curs_set()`, these routines always return `OK`. `curs_set()` returns the previous cursor state, or `ERR` if the requested *visibility* is not supported.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_initscr(3CURSES)`, `curs_outopts(3CURSES)`, `curs_refresh(3CURSES)`, `curs_scr_dump(3CURSES)`, `curs_slk(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `getsyx()` is a macro, so an ampersand (&) is not necessary before the variables *y* and *x*.

| NAME | def_prog_mode, def_shell_mode, reset_prog_mode, reset_shell_mode – save/restore terminal modes | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int def_prog_mode(void); int def_shell_mode(void); int reset_prog_mode(void); int reset_shell_mode(void);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>def_prog_mode()</code> and <code>def_shell_mode()</code> functions save the current terminal modes as "program" (within X/Open Curses) or "shell" (outside X/Open Curses). The modes are saved automatically by <code>initscr(3XCURSES)</code>, <code>newterm(3XCURSES)</code>, and <code>setupterm(3XCURSES)</code>.</p> <p>The <code>reset_prog_mode()</code> and <code>reset_shell_mode()</code> functions reset the current terminal modes to "program" (within X/Open Curses) or "shell" (outside X/Open Curses). The <code>endwin(3XCURSES)</code> function automatically calls the <code>reset_shell_mode()</code> function and the <code>doupdate(3XCURSES)</code> function calls the <code>reset_prog_mode()</code> function after calling <code>endwin()</code>.</p> | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>endwin(3XCURSES)</code> , <code>initscr(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>newterm(3XCURSES)</code> , <code>setupterm(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

def_shell_mode(3CURSES)

| | |
|--------------------|--|
| NAME | <code> curs_kernel, def_prog_mode, def_shell_mode, reset_prog_mode, reset_shell_mode, resetty, savetty, getsyx, setsyx, ripoffline, curs_set, napms</code> – low-level curses routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int def_prog_mode(void); int def_shell_mode(void); int reset_prog_mode(void); int reset_shell_mode(void); int resetty(void); int savetty(void); int getsyx(int y, int x); int setsyx(int y, int x); int ripoffline(int line, int (*init)(WINDOW *, int)); int curs_set(int visibility); int napms(int ms);</pre> |
| DESCRIPTION | <p>The following routines give low-level access to various curses functionality. These routines typically are used inside library routines.</p> <p>The <code>def_prog_mode()</code> and <code>def_shell_mode()</code> routines save the current terminal modes as the “program” (in curses) or “shell” (not in curses) state for use by the <code>reset_prog_mode()</code> and <code>reset_shell_mode()</code> routines. This is done automatically by <code>initscr()</code>.</p> <p>The <code>reset_prog_mode()</code> and <code>reset_shell_mode()</code> routines restore the terminal to “program” (in curses) or “shell” (out of curses) state. These are done automatically by <code>endwin()</code> and, after an <code>endwin()</code>, by <code>doupdate()</code>, so they normally are not called.</p> <p>The <code>resetty()</code> and <code>savetty()</code> routines save and restore the state of the terminal modes. <code>savetty()</code> saves the current state in a buffer and <code>resetty()</code> restores the state to what it was at the last call to <code>savetty()</code>.</p> <p>With the <code>getsyx()</code> routine, the current coordinates of the virtual screen cursor are returned in <code>y</code> and <code>x</code>. If <code>leaveok()</code> is currently TRUE, then <code>-1,-1</code> is returned. If lines have been removed from the top of the screen, using <code>ripoffline()</code>, <code>y</code> and <code>x</code> include these lines; therefore, <code>y</code> and <code>x</code> should be used only as arguments for <code>setsyx()</code>.</p> <p>With the <code>setsyx()</code> routine, the virtual screen cursor is set to <code>y, x</code>. If <code>y</code> and <code>x</code> are both <code>-1</code>, then <code>leaveok()</code> is set. The two routines <code>getsyx()</code> and <code>setsyx()</code> are designed to be used by a library routine, which manipulates curses windows but does not</p> |

def_shell_mode(3CURSES)

want to change the current position of the program's cursor. The library routine would call `getsyx()` at the beginning, do its manipulation of its own windows, do a `wnoutrefresh()` on its windows, call `setsyx()`, and then call `doupdate()`.

The `riporffline()` routine provides access to the same facility that `slk_init()` (see `curs_slk(3CURSES)`) uses to reduce the size of the screen. `riporffline()` must be called before `initscr()` or `newterm()` is called. If `line` is positive, a line is removed from the top of `stdscr()`; if `line` is negative, a line is removed from the bottom. When this is done inside `initscr()`, the routine `init()` (supplied by the user) is called with two arguments: a window pointer to the one-line window that has been allocated and an integer with the number of columns in the window. Inside this initialization routine, the integer variables `LINES` and `COLS` (defined in `<curses.h>`) are not guaranteed to be accurate and `wrefresh()` or `doupdate()` must not be called. It is allowable to call `wnoutrefresh()` during the initialization routine.

`riporffline()` can be called up to five times before calling `initscr()` or `newterm()`.

With the `curs_set()` routine, the cursor state is set to invisible, normal, or very visible for *visibility* equal to 0, 1, or 2 respectively. If the terminal supports the *visibility* requested, the previous *cursor* state is returned; otherwise, `ERR` is returned.

The `napms()` routine is used to sleep for *ms* milliseconds.

RETURN VALUES

Except for `curs_set()`, these routines always return `OK`. `curs_set()` returns the previous cursor state, or `ERR` if the requested *visibility* is not supported.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_initscr(3CURSES)`, `curs_outopts(3CURSES)`, `curs_refresh(3CURSES)`, `curs_scr_dump(3CURSES)`, `curs_slk(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `getsyx()` is a macro, so an ampersand (&) is not necessary before the variables *y* and *x*.

def_shell_mode(3XCURSES)

| NAME | def_prog_mode, def_shell_mode, reset_prog_mode, reset_shell_mode – save/restore terminal modes | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int def_prog_mode(void); int def_shell_mode(void); int reset_prog_mode(void); int reset_shell_mode(void);</pre> | | | | | | |
| DESCRIPTION | <p>The def_prog_mode() and def_shell_mode() functions save the current terminal modes as "program" (within X/Open Curses) or "shell" (outside X/Open Curses). The modes are saved automatically by initscr(3XCURSES), newterm(3XCURSES), and setupterm(3XCURSES).</p> <p>The reset_prog_mode() and reset_shell_mode() functions reset the current terminal modes to "program" (within X/Open Curses) or "shell" (outside X/Open Curses). The endwin(3XCURSES) function automatically calls the reset_shell_mode() function and the doupdate(3XCURSES) function calls the reset_prog_mode() function after calling endwin().</p> | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Standard</td></tr><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | endwin(3XCURSES), initscr(3XCURSES), libcurses(3XCURSES), newterm(3XCURSES), setupterm(3XCURSES), attributes(5), standards(5) | | | | | | |

delay_output(3CURSES)

| | |
|-------------|--|
| NAME | curs_util, unctrl, keyname, filter, use_env, putwin, getwin, delay_output, flushinp – curses miscellaneous utility routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> char *unctrl (chtype <i>c</i>); char *keyname (int <i>c</i>); int filter (void); void use_env (char <i>bool</i>); int putwin (WINDOW *<i>win</i>, FILE *<i>filep</i>); WINDOW *getwin (FILE *<i>filep</i>); int delay_output (int <i>ms</i>); int flushinp (void);</pre> |
| DESCRIPTION | <p>The unctrl () macro expands to a character string which is a printable representation of the character <i>c</i>. Control characters are displayed in the ^X notation. Printing characters are displayed as is.</p> <p>With the keyname () routine, a character string corresponding to the key <i>c</i> is returned.</p> <p>The filter () routine, if used, is called before initscr () or newterm () are called. It makes curses think that there is a one-line screen. curses does not use any terminal capabilities that assume that they know on what line of the screen the cursor is positioned.</p> <p>The use_env () routine, if used, is called before initscr () or newterm () are called. When called with FALSE as an argument, the values of lines and columns specified in the terminfo database will be used, even if environment variables LINES and COLUMNS (used by default) are set, or if curses is running in a window (in which case default behavior would be to use the window size if LINES and COLUMNS are not set).</p> <p>With the putwin () routine, all data associated with window <i>win</i> is written into the file to which <i>filep</i> points. This information can be later retrieved using the getwin () function.</p> <p>The getwin () routine reads window related data stored in the file by putwin (). The routine then creates and initializes a new window using that data. It returns a pointer to the new window.</p> <p>The delay_output () routine inserts an <i>ms</i> millisecond pause in output. This routine should not be used extensively because padding characters are used rather than a CPU pause.</p> <p>The flushinp () routine throws away any typeahead that has been typed by the user and has not yet been read by the program.</p> |

delay_output(3CURSES)

RETURN VALUES Except for `flushinp()`, routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`flushinp()` always returns `OK`.

Routines that return pointers return `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_initscr(3CURSES)`, `curl_scr_dump(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `unctrl()` is a macro, which is defined in `<unctrl.h>`.

delay_output(3XCURSES)

NAME delay_output – delays output

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

int delay_output (int ms);
```

DESCRIPTION The delay_output () function delays output for *ms* milliseconds by inserting pad characters in the output stream.

PARAMETERS *ms* Is the number of milliseconds to delay the output.

RETURN VALUES On success, the delay_output () function returns OK. Otherwise, it returns ERR.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO libcurses(3XCURSES), napms(3XCURSES), attributes(5), standards(5)

delch(3CURSES)

NAME curs_delch, delch, wdelch, mvdelch, mvwdelch – delete character under cursor in a curses window

SYNOPSIS

```
cc [ flag ... ] file ... -lcurses [ library ... ]
#include <curses.h>

int delch(void);
int wdelch(WINDOW *win);
int mvdelch(int y, int x);
int mvwdelch(WINDOW *win, int y, int x);
```

DESCRIPTION With these routines the character under the cursor in the window is deleted; all characters to the right of the cursor on the same line are moved to the left one position and the last character on the line is filled with a blank. The cursor position does not change (after moving to *y, x*, if specified). This does not imply use of the hardware delete character feature.

RETURN VALUES All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), attributes(5)

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `delch()`, `mvdelch()`, and `mvwdelch()` may be macros.

| NAME | delch, mvdelch, mvwdelch, wdelch – remove a character | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int delch(void); int mvdelch(int y, int x); int mvwdelch(WINDOW *win, int y, int x); int wdelch(WINDOW *win);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>delch()</code> and <code>wdelch()</code> functions delete the character at the current cursor position from <code>stdscr</code> and <code>win</code>, respectively. All remaining characters after cursor through to the end of the line are shifted one character towards the start of the line. The last character on the line becomes a space; characters on other lines are not affected.</p> <p>The <code>mvdelch()</code> and <code>mvwdelch()</code> functions delete the character at the position specified by the <code>x</code> and <code>y</code> parameters; the former deletes the character from <code>stdscr</code>; the latter from <code>win</code>.</p> | | | | | | |
| PARAMETERS | <p><code>y</code> Is the <code>y</code> (row) coordinate of the position of the character to be removed.</p> <p><code>x</code> Is the <code>x</code> (column) coordinate of the position of the character to be removed.</p> <p><code>win</code> Is a pointer to the window containing the character to be removed.</p> | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>bkgdset(3XCURSES)</code> , <code>insch(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

del_curterm(3CURSES)

| | |
|--------------------|--|
| NAME | <code> curs_terminfo, setupterm, setterm, set_curterm, del_curterm, restartterm, tparm, tputs, putp, vidputs, vidattr, mvcur, tigetflag, tigetnum, tigetstr</code> – curses interfaces to terminfo database |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> #include <term.h> int setupterm(char *term, int fildes, int *errret); int setterm(char *term); int set_curterm(TERMINAL *nterm); int del_curterm(TERMINAL *oterm); int restartterm(char *term, int fildes, int *errret); char *tparm(char *str, long int p1, long int p2, long int p3, long int p4, long int p5, long int p6, long int p7, long int p8, long int p9); int tputs(char *str, int affcnt, int (*putc)(char)); int putp(char *str); int vidputs(chtype attrs, int (*putc)(char)); int vidattr(chtype attrs); int mvcur(int oldrow, int oldcol, int newrow, int newcol); int tigetflag(char *capname); int tigetnum(char *capname); char *tigetstr(char *capname);</pre> |
| DESCRIPTION | <p>These low-level routines must be called by programs that have to deal directly with the <i>terminfo</i> database to handle certain terminal capabilities, such as programming function keys. For all other functionality, curses routines are more suitable and their use is recommended.</p> <p>Initially, <code>setupterm()</code> should be called. Note that <code>setupterm()</code> is automatically called by <code>initscr()</code> and <code>newterm()</code>. This defines the set of terminal-dependent variables (listed in <code>terminfo(4)</code>). The <i>terminfo</i> variables <code>lines</code> and <code>columns</code> are initialized by <code>setupterm()</code> as follows: If <code>use_env(FALSE)</code> has been called, values for <code>lines</code> and <code>columns</code> specified in <i>terminfo</i> are used. Otherwise, if the environment variables <code>LINES</code> and <code>COLUMNS</code> exist, their values are used. If these environment variables do not exist and the program is running in a window, the current window size is used. Otherwise, if the environment variables do not exist, the values for <code>lines</code> and <code>columns</code> specified in the <i>terminfo</i> database are used.</p> <p>The headers <code><curses.h></code> and <code><term.h></code> should be included (in this order) to get the definitions for these strings, numbers, and flags. Parameterized strings should be passed through <code>tparm()</code> to instantiate them. All <i>terminfo</i> strings (including the output</p> |

del_curterm(3CURSES)

of `tparm()` should be printed with `tputs()` or `putp()`. Call the `reset_shell_mode()` routine to restore the tty modes before exiting (see `curs_kernel(3CURSES)`). Programs which use cursor addressing should output `enter_ca_mode` upon startup and should output `exit_ca_mode` before exiting. Programs desiring shell escapes should call `reset_shell_mode` and output `exit_ca_mode` before the shell is called and should output `enter_ca_mode` and call `reset_prog_mode` after returning from the shell.

The `setupterm()` routine reads in the *terminfo* database, initializing the *terminfo* structures, but does not set up the output virtualization structures used by *curses*. The terminal type is the character string *term*; if *term* is null, the environment variable `TERM` is used. All output is to file descriptor *fildev* which is initialized for output. If *errret* is not null, then `setupterm()` returns `OK` or `ERR` and stores a status value in the integer pointed to by *errret*. A status of 1 in *errret* is normal, 0 means that the terminal could not be found, and -1 means that the *terminfo* database could not be found. If *errret* is null, `setupterm()` prints an error message upon finding an error and exits. Thus, the simplest call is:

```
setupterm((char *)0, 1, (int *)0);
```

which uses all the defaults and sends the output to `stdout`.

The `setterm()` routine is being replaced by `setupterm()`. The call:

```
setupterm(term, 1, (int *)0)
```

provides the same functionality as `setterm(term)`. The `setterm()` routine is included here for compatibility and is supported at Level 2.

The `set_curterm()` routine sets the variable `cur_term` to *nterm*, and makes all of the *terminfo* boolean, numeric, and string variables use the values from *nterm*.

The `del_curterm()` routine frees the space pointed to by *oterm* and makes it available for further use. If *oterm* is the same as `cur_term`, references to any of the *terminfo* boolean, numeric, and string variables thereafter may refer to invalid memory locations until another `setupterm()` has been called.

The `restartterm()` routine is similar to `setupterm()` and `initscr()`, except that it is called after restoring memory to a previous state. It assumes that the windows and the input and output options are the same as when memory was saved, but the terminal type and baud rate may be different.

The `tparm()` routine instantiates the string *str* with parameters *pi*. A pointer is returned to the result of *str* with the parameters applied.

The `tputs()` routine applies padding information to the string *str* and outputs it. The *str* must be a *terminfo* string variable or the return value from `tparm()`, `tgetstr()`, or `tgoto()`. *affcnt* is the number of lines affected, or 1 if not applicable. *putc* is a `putchar()`-like routine to which the characters are passed, one at a time.

del_curterm(3CURSES)

The `putp()` routine calls `tputs(str, 1, putchar)`. Note that the output of `putpA()` always goes to `stdout`, not to the *files* specified in `setupterm()`.

The `vidputs()` routine displays the string on the terminal in the video attribute mode *attrs*, which is any combination of the attributes listed in `curses(3CURSES)`. The characters are passed to the `putchar()`-like routine `putc()`.

The `vidattr()` routine is like the `vidputs()` routine, except that it outputs through `putchar()`.

The `mvcur()` routine provides low-level cursor motion.

The `tigetflag()`, `tigetnum()` and `tigetstr()` routines return the value of the capability corresponding to the *terminfo capname* passed to them, such as `xenl`.

With the `tigetflag()` routine, the value `-1` is returned if *capname* is not a boolean capability.

With the `tigetnum()` routine, the value `-2` is returned if *capname* is not a numeric capability.

With the `tigetstr()` routine, the value `(char *)-1` is returned if *capname* is not a string capability.

The *capname* for each capability is given in the table column entitled *capname code* in the capabilities section of `terminfo(4)`.

```
char *boolnames, *boolcodes, *boolfnames
char *numnames, *numcodes, *numfnames
char *strnames, *strcodes, *strfnames
```

These null-terminated arrays contain the *capnames*, the *termcap* codes, and the full C names, for each of the *terminfo* variables.

RETURN VALUES

All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

Routines that return pointers always return `NULL` on error.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_initscr(3CURSES)`, `curs_kernel(3CURSES)`, `curs_termcap(3CURSES)`, `curses(3CURSES)`, `putc(3C)`, `terminfo(4)`, `attributes(5)`

`del_curterm(3CURSES)`

NOTES | The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

The `setupterm()` routine should be used in place of `setterm()`.

Note that `vidattr()` and `vidputs()` may be macros.

del_curterm(3XCURSES)

| NAME | del_curterm, restartterm, set_curterm, setupterm – interfaces to the terminfo database | | | | | | | | |
|--------------------|--|-------|-------------|---|--------|---|-----------------------------|----|--------------------------------------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <term.h> int del_curterm(TERMINAL *oterm); int restartterm(char *term, int fildes, int *errret); TERMINAL *set_curterm(TERMINAL *nterm); int setupterm(char *term, int fildes, int *errret);</pre> | | | | | | | | |
| DESCRIPTION | <p>Within X/Open Curses, the <code>setupterm()</code> function is automatically called by the <code>initscr(3XC)</code> and <code>newterm(3XC)</code> functions. This function can be also be used outside of X/Open Curses when a program has to deal directly with the <code>terminfo</code> database to handle certain terminal capabilities. The use of appropriate X/Open Curses functions is recommended in all other situations.</p> <p>The <code>setupterm()</code> function loads terminal-dependent variables for the <code>terminfo</code> layer of X/Open Curses. The <code>setupterm()</code> function initializes the <code>terminfo</code> variables <code>lines</code> and <code>columns</code> such that if <code>use_env(FALSE)</code> has been called, the <code>terminfo</code> values assigned in the database are used regardless of the environmental variables <code>LINES</code> and <code>COLUMNS</code> or the program's window dimensions; when <code>use_env(TRUE)</code> has been called, which is the default, the environment variables <code>LINES</code> and <code>COLUMNS</code> are used, if they exist. If the environment variables do not exist and the program is running in a window, the current window size is used.</p> <p>The <code>term</code> parameter of <code>setupterm()</code> specifies the terminal; if null, terminal type is taken from the <code>TERM</code> environment variable. All output is sent to <code>fildes</code> which is initialized for output. If <code>errret</code> is not null, <code>OK</code> or <code>ERR</code> is returned and a status value is stored in the integer pointed to by <code>errret</code>. The following status values may be returned:</p> <table border="1"><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>1</td><td>Normal</td></tr><tr><td>0</td><td>Terminal could not be found</td></tr><tr><td>-1</td><td>terminfo database could not be found</td></tr></tbody></table> <p>If <code>errret</code> is null, an error message is printed, and the <code>setupterm()</code> function calls the <code>exit()</code> function with a non-zero parameter.</p> | Value | Description | 1 | Normal | 0 | Terminal could not be found | -1 | terminfo database could not be found |
| Value | Description | | | | | | | | |
| 1 | Normal | | | | | | | | |
| 0 | Terminal could not be found | | | | | | | | |
| -1 | terminfo database could not be found | | | | | | | | |

del_curterm(3XCURSES)

The set_curterm() function sets the cur_term variable to nterm. The values from nterm as well as other state information for the terminal are used by X/Open Curses functions such as beep(3XCURSES), flash(3XCURSES), mvcur(3XCURSES), tigetflag(3XCURSES), tigetstr(3XCURSES), and tigetnum(3XCURSES).

The del_curterm() function frees the space pointed to by oterm. If oterm and the cur_term variable are the same, all Boolean, numeric, or string terminfo variables will refer to invalid memory locations until you call setupterm() and specify a new terminal type.

The restartterm() function assumes that a call to setupterm() has already been made (probably from initscr() or newterm()). It allows you to specify a new terminal type in term and updates the data returned by baudrate(3XCURSES) based on fildes. Other information created by the initscr(), newterm(), and setupterm() functions is preserved.

PARAMETERS

- oterm* Is the terminal type for which to free space.
- term* Is the terminal type for which variables are set.
- fildes* Is a file descriptor initialized for output.
- errret* Is a pointer to an integer in which the status value is stored.
- nterm* Is the new terminal to become the current terminal.

RETURN VALUES

On success, the set_curterm() function returns the previous value of cur_term. Otherwise, it returns a null pointer.

On success, the other functions return OK. Otherwise, they return ERR.

ERRORS

None.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO

baudrate(3XCURSES), beep(3XCURSES), initscr(3XCURSES), libcurses(3XCURSES), mvcur(3XCURSES), tigetflag(3XCURSES), use_env(3XCURSES), attributes(5), standards(5)

deleteln(3CURSES)

| NAME | <code> curs_deleteln, deleteln, wdeleteln, insdelln, winsdelln, insertln, winsertln</code> – delete and insert lines in a curses window | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int deleteln(void); int wdeleteln(WINDOW *<i>win</i>); int insdelln(int <i>n</i>); int winsdelln(WINDOW *<i>win</i>, int <i>n</i>); int insertln(void); int winsertln(WINDOW *<i>win</i>);</pre> | | | | |
| DESCRIPTION | <p>With the <code>deleteln()</code> and <code>wdeleteln()</code> routines, the line under the cursor in the window is deleted; all lines below the current line are moved up one line. The bottom line of the window is cleared. The cursor position does not change. This does not imply use of a hardware delete line feature.</p> <p>With the <code>insdelln()</code> and <code>winsdelln()</code> routines, for positive <i>n</i>, insert <i>n</i> lines into the specified window above the current line. The <i>n</i> bottom lines are lost. For negative <i>n</i>, delete <i>n</i> lines (starting with the one under the cursor), and move the remaining lines up. The bottom <i>n</i> lines are cleared. The current cursor position remains the same.</p> <p>With the <code>insertln()</code> and <code>winsertln()</code> routines, a blank line is inserted above the current line and the bottom line is lost. This does not imply use of a hardware insert line feature.</p> | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . Note that all but <code>winsdelln()</code> may be macros. | | | | |

NAME deleteln, wdeleteln – remove a line

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

int deleteln(void);

int wdeleteln(WINDOW *win);
```

DESCRIPTION The `deleteln()` and `wdeleteln()` functions delete the line containing the cursor from `stdscr` and `win`, respectively. All lines below the one deleted are moved up one line. The last line of the window becomes blank. The position of the cursor is unchanged.

PARAMETERS `win` Is a pointer to the window from which the line is removed.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `bkgdset(3XCURSES)`, `insdelln(3XCURSES)`, `insertln(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

del_panel(3CURSES)

NAME panel_new, new_panel, del_panel – create and destroy panels

SYNOPSIS

```
cc [ flag ... ] file ... -lpanel -lcurses [ library .. ]
#include <panel.h>

PANEL *new_panel (WINDOW *win) ;
int del_panel (PANEL *panel) ;
```

DESCRIPTION new_panel () creates a new panel associated with *win* and returns the panel pointer. The new panel is placed on top of the panel deck.

del_panel () destroys *panel*, but not its associated window.

RETURN VALUES new_panel () returns NULL if an error occurs.

del_win () returns OK if successful, ERR otherwise.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), panel_update(3CURSES), panels(3CURSES), attributes(5)

NOTES The header <panel.h> automatically includes the header <curses.h>.

| | |
|--------------------|--|
| NAME | curs_initscr, initscr, newterm, endwin, isendwin, set_term, delscreen – curses screen initialization and manipulation routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> WINDOW *initscr(void); int endwin(void); int isendwin(void); SCREEN *newterm(char *<i>type</i>, FILE *<i>outfd</i>, FILE *<i>infd</i>); SCREEN *set_term(SCREEN *<i>new</i>); void delscreen(SCREEN * <i>sp</i>);</pre> |
| DESCRIPTION | <p><code>initscr()</code> is almost always the first routine that should be called (the exceptions are <code>slk_init()</code>, <code>filter()</code>, <code>ripoffline()</code>, <code>use_env()</code> and, for multiple-terminal applications, <code>newterm()</code>.) This determines the terminal type and initializes all curses data structures. <code>initscr()</code> also causes the first call to <code>refresh()</code> to clear the screen. If errors occur, <code>initscr()</code> writes an appropriate error message to standard error and exits; otherwise, a pointer is returned to <code>stdscr()</code>. If the program needs an indication of error conditions, <code>newterm()</code> should be used instead of <code>initscr()</code>; <code>initscr()</code> should only be called once per application.</p> <p>A program that outputs to more than one terminal should use the <code>newterm()</code> routine for each terminal instead of <code>initscr()</code>. A program that needs an indication of error conditions, so it can continue to run in a line-oriented mode if the terminal cannot support a screen-oriented program, would also use this routine. The routine <code>newterm()</code> should be called once for each terminal. It returns a variable of type <code>SCREEN *</code> which should be saved as a reference to that terminal. The arguments are the <i>type</i> of the terminal to be used in place of <code>\$TERM</code>, a file pointer for output to the terminal, and another file pointer for input from the terminal (if <i>type</i> is <code>NULL</code>, <code>\$TERM</code> will be used). The program must also call <code>endwin()</code> for each terminal being used before exiting from curses. If <code>newterm()</code> is called more than once for the same terminal, the first terminal referred to must be the last one for which <code>endwin()</code> is called.</p> <p>A program should always call <code>endwin()</code> before exiting or escaping from curses mode temporarily. This routine restores tty modes, moves the cursor to the lower left-hand corner of the screen and resets the terminal into the proper non-visual mode. Calling <code>refresh()</code> or <code>doupdate()</code> after a temporary escape causes the program to resume visual mode.</p> <p>The <code>isendwin()</code> routine returns <code>TRUE</code> if <code>endwin()</code> has been called without any subsequent calls to <code>wrefresh()</code>, and <code>FALSE</code> otherwise.</p> |

delscreen(3CURSES)

The `set_term()` routine is used to switch between different terminals. The screen reference `new` becomes the new current terminal. The previous terminal is returned by the routine. This is the only routine which manipulates `SCREEN` pointers; all other routines affect only the current terminal.

The `delscreen()` routine frees storage associated with the `SCREEN` data structure. The `endwin()` routine does not do this, so `delscreen()` should be called after `endwin()` if a particular `SCREEN` is no longer needed.

RETURN VALUES `endwin()` returns the integer `ERR` upon failure and `OK` upon successful completion.

Routines that return pointers always return `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_kernel(3CURSES)`, `curl_refresh(3CURSES)`, `curl_slk(3CURSES)`, `curl_util(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `initscr()` and `newterm()` may be macros.

NAME | delscreen – free space associated with the SCREEN data structure

SYNOPSIS |

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

void delscreen(SCREEN *sp);
```

DESCRIPTION | The `delscreen()` function frees space associated with the SCREEN data structure. This function should be called after `endwin(3XCURSES)` if a SCREEN data structure is no longer needed.

PARAMETERS | `sp` Is a pointer to the screen structure for which to free space.

RETURN VALUES | The `delscreen()` function does not return a value.

ERRORS | None.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | `endwin(3XCURSES)`, `initscr(3XCURSES)`, `libcurses(3XCURSES)`, `newterm(3XCURSES)`, `attributes(5)`, `standards(5)`

delwin(3CURSES)

| | |
|--------------------|--|
| NAME | <code>curs_window</code> , <code>newwin</code> , <code>delwin</code> , <code>mvwin</code> , <code>subwin</code> , <code>derwin</code> , <code>mvderwin</code> , <code>dupwin</code> , <code>wsyncup</code> , <code>syncok</code> , <code>wcursyncup</code> , <code>wsyncdown</code> – create curses windows |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> WINDOW *newwin(int nlines, int ncols, int begin_y, int begin_x); int delwin(WINDOW *win); int mvwin(WINDOW *win, int y, int x); WINDOW *subwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); WINDOW *derwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); int mvderwin(WINDOW *win, int par_y, int par_x); WINDOW *dupwin(WINDOW *win); void wsyncup(WINDOW *win); int syncok(WINDOW *win, bool bf); void wcursyncup(WINDOW *win); void wsyncdown(WINDOW *win);</pre> |
| DESCRIPTION | <p>The <code>newwin()</code> routine creates and returns a pointer to a new window with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. The upper left-hand corner of the window is at line <i>begin_y</i>, column <i>begin_x</i>. If either <i>nlines</i> or <i>ncols</i> is zero, they default to <code>LINES</code> — <i>begin_y</i> and <code>COLS</code> — <i>begin_x</i>. A new full-screen window is created by calling <code>newwin(0, 0, 0, 0)</code>.</p> <p>The <code>delwin()</code> routine deletes the named window, freeing all memory associated with it. Subwindows must be deleted before the main window can be deleted.</p> <p>The <code>mvwin()</code> routine moves the window so that the upper left-hand corner is at position (<i>x</i>, <i>y</i>). If the move would cause the window to be off the screen, it is an error and the window is not moved. Moving subwindows is allowed, but should be avoided.</p> <p>The <code>subwin()</code> routine creates and returns a pointer to a new window with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. The window is at position (<i>begin_y</i>, <i>begin_x</i>) on the screen. (This position is relative to the screen, and not to the window <i>orig</i>.) The window is made in the middle of the window <i>orig</i>, so that changes made to one window will affect both windows. The subwindow shares memory with the window <i>orig</i>. When using this routine, it is necessary to call <code>touchwin()</code> or <code>touchline()</code> on <i>orig</i> before calling <code>wrefresh()</code> on the subwindow.</p> <p>The <code>derwin()</code> routine is the same as <code>subwin()</code>, except that <i>begin_y</i> and <i>begin_x</i> are relative to the origin of the window <i>orig</i> rather than the screen. There is no difference between the subwindows and the derived windows.</p> |

delwin(3CURSES)

The `mvderwin()` routine moves a derived window (or subwindow) inside its parent window. The screen-relative parameters of the window are not changed. This routine is used to display different parts of the parent window at the same physical position on the screen.

The `dupwin()` routine creates an exact duplicate of the window *win*.

Each `curses` window maintains two data structures: the character image structure and the status structure. The character image structure is shared among all windows in the window hierarchy (that is, the window with all subwindows). The status structure, which contains information about individual line changes in the window, is private to each window. The routine `wrefresh()` uses the status data structure when performing screen updating. Since status structures are not shared, changes made to one window in the hierarchy may not be properly reflected on the screen.

The routine `wsyncup()` causes the changes in the status structure of a window to be reflected in the status structures of its ancestors. If `syncok()` is called with second argument `TRUE` then `wsyncup()` is called automatically whenever there is a change in the window.

The routine `wcursyncup()` updates the current cursor position of all the ancestors of the window to reflect the current cursor position of the window.

The routine `wsyncdown()` updates the status structure of the window to reflect the changes in the status structures of its ancestors. Applications seldom call this routine because it is called automatically by `wrefresh()`.

RETURN VALUES

Routines that return an integer return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`delwin()` returns the integer `ERR` upon failure and `OK` upon successful completion.

Routines that return pointers return `NULL` on error.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_refresh(3CURSES)`, `curs_touch(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

If many small changes are made to the window, the `wsyncup()` option could degrade performance.

delwin(3CURSES)

Note that `syncok()` may be a macro.

| NAME | delwin – delete a window | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int delwin(WINDOW *win);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>delwin()</code> function deletes the specified window, freeing up the memory associated with it.</p> <p>Deleting a parent window without deleting its subwindows and then trying to manipulate the subwindows will have undefined results.</p> | | | | | | |
| PARAMETERS | <i>win</i> Is a pointer to the window that is to be deleted. | | | | | | |
| RETURN VALUES | On success, this functions returns OK. Otherwise, it returns ERR. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>derwin(3XCURSES)</code> , <code>dupwin(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

derwin(3CURSES)

| | |
|--------------------|--|
| NAME | <code>curs_window</code> , <code>newwin</code> , <code>delwin</code> , <code>mvwin</code> , <code>subwin</code> , <code>derwin</code> , <code>mvderwin</code> , <code>dupwin</code> , <code>wsyncup</code> , <code>syncok</code> , <code>wcursyncup</code> , <code>wsyncdown</code> – create curses windows |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> WINDOW *newwin(int nlines, int ncols, int begin_y, int begin_x); int delwin(WINDOW *win); int mvwin(WINDOW *win, int y, int x); WINDOW *subwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); WINDOW *derwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); int mvderwin(WINDOW *win, int par_y, int par_x); WINDOW *dupwin(WINDOW *win); void wsyncup(WINDOW *win); int syncok(WINDOW *win, bool bf); void wcursyncup(WINDOW *win); void wsyncdown(WINDOW *win);</pre> |
| DESCRIPTION | <p>The <code>newwin()</code> routine creates and returns a pointer to a new window with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. The upper left-hand corner of the window is at line <i>begin_y</i>, column <i>begin_x</i>. If either <i>nlines</i> or <i>ncols</i> is zero, they default to <code>LINES</code> — <i>begin_y</i> and <code>COLS</code> — <i>begin_x</i>. A new full-screen window is created by calling <code>newwin(0, 0, 0, 0)</code>.</p> <p>The <code>delwin()</code> routine deletes the named window, freeing all memory associated with it. Subwindows must be deleted before the main window can be deleted.</p> <p>The <code>mvwin()</code> routine moves the window so that the upper left-hand corner is at position (<i>x</i>, <i>y</i>). If the move would cause the window to be off the screen, it is an error and the window is not moved. Moving subwindows is allowed, but should be avoided.</p> <p>The <code>subwin()</code> routine creates and returns a pointer to a new window with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. The window is at position (<i>begin_y</i>, <i>begin_x</i>) on the screen. (This position is relative to the screen, and not to the window <i>orig</i>.) The window is made in the middle of the window <i>orig</i>, so that changes made to one window will affect both windows. The subwindow shares memory with the window <i>orig</i>. When using this routine, it is necessary to call <code>touchwin()</code> or <code>touchline()</code> on <i>orig</i> before calling <code>wrefresh()</code> on the subwindow.</p> <p>The <code>derwin()</code> routine is the same as <code>subwin()</code>, except that <i>begin_y</i> and <i>begin_x</i> are relative to the origin of the window <i>orig</i> rather than the screen. There is no difference between the subwindows and the derived windows.</p> |

The `mvderwin()` routine moves a derived window (or subwindow) inside its parent window. The screen-relative parameters of the window are not changed. This routine is used to display different parts of the parent window at the same physical position on the screen.

The `dupwin()` routine creates an exact duplicate of the window *win*.

Each `curses` window maintains two data structures: the character image structure and the status structure. The character image structure is shared among all windows in the window hierarchy (that is, the window with all subwindows). The status structure, which contains information about individual line changes in the window, is private to each window. The routine `wrefresh()` uses the status data structure when performing screen updating. Since status structures are not shared, changes made to one window in the hierarchy may not be properly reflected on the screen.

The routine `wsyncup()` causes the changes in the status structure of a window to be reflected in the status structures of its ancestors. If `syncok()` is called with second argument `TRUE` then `wsyncup()` is called automatically whenever there is a change in the window.

The routine `wcursyncup()` updates the current cursor position of all the ancestors of the window to reflect the current cursor position of the window.

The routine `wsyncdown()` updates the status structure of the window to reflect the changes in the status structures of its ancestors. Applications seldom call this routine because it is called automatically by `wrefresh()`.

RETURN VALUES

Routines that return an integer return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`delwin()` returns the integer `ERR` upon failure and `OK` upon successful completion.

Routines that return pointers return `NULL` on error.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_refresh(3CURSES)`, `curs_touch(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

If many small changes are made to the window, the `wsyncup()` option could degrade performance.

derwin(3CURSES)

Note that `syncok()` may be a macro.

| | | | | | | | | | |
|--------------------|--|-------------|--|---------------|--|--------------|--|----------------|---|
| NAME | derwin, newwin, subwin – create a new window or subwindow | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> WINDOW *derwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); WINDOW *newwin(int nlines, int ncols, int begin_y, int begin_x); WINDOW *subwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x);</pre> | | | | | | | | |
| DESCRIPTION | <p>The <code>derwin()</code> function creates a subwindow within window <i>orig</i>, with the specified number of lines and columns, and upper left corner positioned at <i>begin_x</i>, <i>begin_y</i> relative to window <i>orig</i>. A pointer to the new window structure is returned.</p> <p>The <code>newwin()</code> function creates a new window with the specified number of lines and columns and upper left corner positioned at <i>begin_x</i>, <i>begin_y</i>. A pointer to the new window structure is returned. A full-screen window can be created by calling <code>newwin(0, 0, 0, 0)</code>.</p> <p>If the number of lines specified is zero, <code>newwin()</code> uses a default value of <code>LINES</code> minus <i>begin_y</i>; if the number of columns specified is zero, <code>newwin()</code> uses the default value of <code>COLS</code> minus <i>begin_x</i>.</p> <p>The <code>subwin()</code> function creates a subwindow within window <i>orig</i>, with the specified number of lines and columns, and upper left corner positioned at <i>begin_x</i>, <i>begin_y</i> (relative to the physical screen, <i>not</i> to window <i>orig</i>). A pointer to the new window structure is returned.</p> <p>The original window and subwindow share character storage of the overlapping area (each window maintains its own pointers, cursor location, and other items). This means that characters and attributes are identical in overlapping areas regardless of which window characters are written to.</p> <p>When using subwindows, it is often necessary to call <code>touchwin(3XCURSES)</code> before <code>wrefresh(3XCURSES)</code> to maintain proper screen contents.</p> | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>orig</i></td> <td>Is a pointer to the parent window for the newly created subwindow.</td> </tr> <tr> <td><i>nlines</i></td> <td>Is the number of lines in the subwindow.</td> </tr> <tr> <td><i>ncols</i></td> <td>Is the number of columns in the subwindow.</td> </tr> <tr> <td><i>begin_y</i></td> <td>Is the y (row) coordinate of the upper left corner of the subwindow, relative to the parent window.</td> </tr> </table> | <i>orig</i> | Is a pointer to the parent window for the newly created subwindow. | <i>nlines</i> | Is the number of lines in the subwindow. | <i>ncols</i> | Is the number of columns in the subwindow. | <i>begin_y</i> | Is the y (row) coordinate of the upper left corner of the subwindow, relative to the parent window. |
| <i>orig</i> | Is a pointer to the parent window for the newly created subwindow. | | | | | | | | |
| <i>nlines</i> | Is the number of lines in the subwindow. | | | | | | | | |
| <i>ncols</i> | Is the number of columns in the subwindow. | | | | | | | | |
| <i>begin_y</i> | Is the y (row) coordinate of the upper left corner of the subwindow, relative to the parent window. | | | | | | | | |

derwin(3XCURSES)

begin_x Is the x (column) coordinate of the upper left corner of the subwindow, relative to the parent window.

RETURN VALUES On success, these functions return a pointer to the newly-created window. Otherwise, they return `ERR`.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `doupdate(3XCURSES)`, `is_linetouched(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|---|
| NAME | curs_refresh, refresh, wrefresh, wnoutrefresh, doupdate, redrawwin, wredrawln – refresh curses windows and lines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int refresh(void); int wrefresh(WINDOW *win); int wnoutrefresh(WINDOW *win); int doupdate(void); int redrawwin(WINDOW *win); int wredrawln(WINDOW *win, int beg_line, int num_lines);</pre> |
| DESCRIPTION | <p>The <code>refresh()</code> and <code>wrefresh()</code> routines (or <code>wnoutrefresh()</code> and <code>doupdate()</code>) must be called to get any output on the terminal, as other routines merely manipulate data structures. The routine <code>wrefresh()</code> copies the named window to the physical terminal screen, taking into account what is already there in order to do optimizations. The <code>refresh()</code> routine is the same, using <code>stdscr</code> as the default window. Unless <code>leaveok()</code> has been enabled, the physical cursor of the terminal is left at the location of the cursor for that window.</p> <p>The <code>wnoutrefresh()</code> and <code>doupdate()</code> routines allow multiple updates with more efficiency than <code>wrefresh()</code> alone. In addition to all the window structures, <code>curses</code> keeps two data structures representing the terminal screen: a physical screen, describing what is actually on the screen, and a virtual screen, describing what the programmer wants to have on the screen.</p> <p>The routine <code>wrefresh()</code> works by first calling <code>wnoutrefresh()</code>, which copies the named window to the virtual screen, and then calling <code>doupdate()</code>, which compares the virtual screen to the physical screen and does the actual update. If the programmer wishes to output several windows at once, a series of calls to <code>wrefresh()</code> results in alternating calls to <code>wnoutrefresh()</code> and <code>doupdate()</code>, causing several bursts of output to the screen. By first calling <code>wnoutrefresh()</code> for each window, it is then possible to call <code>doupdate()</code> once, resulting in only one burst of output, with fewer total characters transmitted and less CPU time used. If the <code>win</code> argument to <code>wrefresh()</code> is the global variable <code>curscr</code>, the screen is immediately cleared and repainted from scratch.</p> <p>The <code>redrawwin()</code> routine indicates to <code>curses</code> that some screen lines are corrupted and should be thrown away before anything is written over them. These routines could be used for programs such as editors, which want a command to redraw some part of the screen or the entire screen. The routine <code>wredrawln()</code> is preferred over <code>redrawwin()</code> where a noisy communication line exists and redrawing the entire window could be subject to even more communication noise. Just redrawing several lines offers the possibility that they would show up unblemished.</p> |

doupdate(3CURSES)

RETURN VALUES All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_ouptops(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `refresh()` and `redrawwin()` may be macros.

| NAME | doupdate, refresh, wnoutrefresh, wrefresh – refresh windows and lines | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int doupdate(void); int refresh(void); int wnoutrefresh(WINDOW *win); int wrefresh(WINDOW *win);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>refresh()</code> and <code>wrefresh()</code> functions copy <code>stdscr</code> and <code>win</code>, respectively, to the terminal screen. These functions call the <code>wnoutrefresh()</code> function to copy the specified window to <code>curscr</code> and the <code>doupdate()</code> function to do the actual update. The physical cursor is mapped to the same position as the logical cursor of the last window to update <code>curscr</code> unless <code>leaveok(3XCURSES)</code> is enabled (in which case, the cursor is placed in a position that X/Open Curses finds convenient).</p> <p>When outputting several windows at once, it is often more efficient to call the <code>wnoutrefresh()</code> and <code>doupdate()</code> functions directly. A call to <code>wnoutrefresh()</code> for each window, followed by only one call to <code>doupdate()</code> to update the screen, results in one burst of output, fewer characters sent, and less CPU time used.</p> <p>If the <code>win</code> parameter to <code>wrefresh()</code> is the global variable <code>curscr</code>, the screen is immediately cleared and repainted from scratch.</p> <p>For details on how the <code>wnoutrefresh()</code> function handles overlapping windows with broad glyphs, see the <i>Overlapping Windows</i> section of the <code>curses(3XCURSES)</code> reference manual page.</p> | | | | | | |
| PARAMETERS | <i>win</i> Is a pointer to the window in which to refresh. | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>clearok(3XCURSES)</code> , <code>curses(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>prefresh(3XCURSES)</code> , <code>redrawwin(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

dup_field(3CURSES)

| NAME | form_field_new, new_field, dup_field, link_field, free_field – create and destroy forms fields | | | | | | | | |
|----------------------|--|----------------|-------------------------------------|-------------|---|----------------|---------------|----------------|---------------------------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> FIELD *new_field(int r, int c, int frow, int fcol, int nrow, int ncol); FIELD *dup_field(FIELD *field, int frow, int fcol); FIELD *link_field(FIELD *field, int frow, int fcol); int free_field(FIELD *field);</pre> | | | | | | | | |
| DESCRIPTION | <p>new_field() creates a new field with <i>r</i> rows and <i>c</i> columns, starting at <i>frow</i>, <i>fcol</i>, in the subwindow of a form. <i>nrow</i> is the number of off-screen rows and <i>nbuf</i> is the number of additional working buffers. This routine returns a pointer to the new field.</p> <p>dup_field() duplicates <i>field</i> at the specified location. All field attributes are duplicated, including the current contents of the field buffers.</p> <p>link_field() also duplicates <i>field</i> at the specified location. However, unlike dup_field(), the new field shares the field buffers with the original field. After creation, the attributes of the new field can be changed without affecting the original field.</p> <p>free_field() frees the storage allocated for <i>field</i>.</p> | | | | | | | | |
| RETURN VALUES | <p>Routines that return pointers return NULL on error. free_field() returns one of the following:</p> <table border="0"> <tr> <td style="padding-right: 20px;">E_OK</td> <td>The function returned successfully.</td> </tr> <tr> <td>E_CONNECTED</td> <td>The field is already connected to a form.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> <tr> <td>E_BAD_ARGUMENT</td> <td>An argument is incorrect.</td> </tr> </table> | E_OK | The function returned successfully. | E_CONNECTED | The field is already connected to a form. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An argument is incorrect. |
| E_OK | The function returned successfully. | | | | | | | | |
| E_CONNECTED | The field is already connected to a form. | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | |
| E_BAD_ARGUMENT | An argument is incorrect. | | | | | | | | |
| ATTRIBUTES | <p>See attributes(5) for descriptions of the following attributes:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | | | |
| MT-Level | Unsafe | | | | | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | | | | | |

| | |
|--------------------|--|
| NAME | curs_window, newwin, delwin, mvwin, subwin, derwin, mvderwin, dupwin, wsyncup, syncok, wcursyncup, wsyncdown – create curses windows |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> WINDOW *newwin(int nlines, int ncols, int begin_y, int begin_x); int delwin(WINDOW *win); int mvwin(WINDOW *win, int y, int x); WINDOW *subwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); WINDOW *derwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); int mvderwin(WINDOW *win, int par_y, int par_x); WINDOW *dupwin(WINDOW *win); void wsyncup(WINDOW *win); int syncok(WINDOW *win, bool bf); void wcursyncup(WINDOW *win); void wsyncdown(WINDOW *win);</pre> |
| DESCRIPTION | <p>The <code>newwin()</code> routine creates and returns a pointer to a new window with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. The upper left-hand corner of the window is at line <i>begin_y</i>, column <i>begin_x</i>. If either <i>nlines</i> or <i>ncols</i> is zero, they default to LINES — <i>begin_y</i> and COLS — <i>begin_x</i>. A new full-screen window is created by calling <code>newwin(0, 0, 0, 0)</code>.</p> <p>The <code>delwin()</code> routine deletes the named window, freeing all memory associated with it. Subwindows must be deleted before the main window can be deleted.</p> <p>The <code>mvwin()</code> routine moves the window so that the upper left-hand corner is at position (<i>x</i>, <i>y</i>). If the move would cause the window to be off the screen, it is an error and the window is not moved. Moving subwindows is allowed, but should be avoided.</p> <p>The <code>subwin()</code> routine creates and returns a pointer to a new window with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. The window is at position (<i>begin_y</i>, <i>begin_x</i>) on the screen. (This position is relative to the screen, and not to the window <i>orig</i>.) The window is made in the middle of the window <i>orig</i>, so that changes made to one window will affect both windows. The subwindow shares memory with the window <i>orig</i>. When using this routine, it is necessary to call <code>touchwin()</code> or <code>touchline()</code> on <i>orig</i> before calling <code>wrefresh()</code> on the subwindow.</p> <p>The <code>derwin()</code> routine is the same as <code>subwin()</code>, except that <i>begin_y</i> and <i>begin_x</i> are relative to the origin of the window <i>orig</i> rather than the screen. There is no difference between the subwindows and the derived windows.</p> |

dupwin(3CURSES)

The `mvderwin()` routine moves a derived window (or subwindow) inside its parent window. The screen-relative parameters of the window are not changed. This routine is used to display different parts of the parent window at the same physical position on the screen.

The `dupwin()` routine creates an exact duplicate of the window *win*.

Each `curses` window maintains two data structures: the character image structure and the status structure. The character image structure is shared among all windows in the window hierarchy (that is, the window with all subwindows). The status structure, which contains information about individual line changes in the window, is private to each window. The routine `wrefresh()` uses the status data structure when performing screen updating. Since status structures are not shared, changes made to one window in the hierarchy may not be properly reflected on the screen.

The routine `wsyncup()` causes the changes in the status structure of a window to be reflected in the status structures of its ancestors. If `syncok()` is called with second argument `TRUE` then `wsyncup()` is called automatically whenever there is a change in the window.

The routine `wcursyncup()` updates the current cursor position of all the ancestors of the window to reflect the current cursor position of the window.

The routine `wsyncdown()` updates the status structure of the window to reflect the changes in the status structures of its ancestors. Applications seldom call this routine because it is called automatically by `wrefresh()`.

RETURN VALUES

Routines that return an integer return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`delwin()` returns the integer `ERR` upon failure and `OK` upon successful completion.

Routines that return pointers return `NULL` on error.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_refresh(3CURSES)`, `curs_touch(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

If many small changes are made to the window, the `wsyncup()` option could degrade performance.

`dupwin(3CURSES)`

Note that `syncok()` may be a macro.

dupwin(3XCURSES)

NAME dupwin – duplicate a window

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

WINDOW *dupwin(WINDOW *win);
```

DESCRIPTION The dupwin() function creates a duplicate of window *win*. A pointer to the new window structure is returned.

PARAMETERS *win* Is a pointer to the window that is to be duplicated.

RETURN VALUES On success, this function returns a pointer to new window structure; otherwise, it returns a null pointer.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO delwin(3XCURSES), derwin(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

NAME form_field_info, field_info, dynamic_field_info – get forms field characteristics

SYNOPSIS

```
cc [ flag... ] file... -lform -lcurses [ library... ]
#include <form.h>

int field_info(FIELD *field, int *rows, int *cols, int *frow, int *fcol,
              int *nrow, int *nbuf);

int dynamic_field_info(FIELD *field, int *drows, int *dcols, int
                      *max);
```

DESCRIPTION

field_info() returns the size, position, and other named field characteristics, as defined in the original call to new_field(), to the locations pointed to by the arguments rows, cols, frow, fcol, nrow, and nbuf.

dynamic_field_info() returns the actual size of the field in the pointer arguments drows, dcols and returns the maximum growth allowed for field in max. If no maximum growth limit is specified for field, max will contain 0. A field can be made dynamic by turning off the field option O_STATIC.

RETURN VALUES These routines return one of the following:

| | |
|----------------|-------------------------------------|
| E_OK | The function returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An argument is incorrect. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), forms(3CURSES), attributes(5)

NOTES The header <form.h> automatically includes the headers <eti.h> and <curses.h>.

echo(3CURSES)

| | |
|--------------------|---|
| NAME | <code>curs_inopts</code> , <code>cbreak</code> , <code>nocbreak</code> , <code>echo</code> , <code>noecho</code> , <code>halfdelay</code> , <code>intrflush</code> , <code>keypad</code> , <code>meta</code> , <code>nodelay</code> , <code>notimeout</code> , <code>raw</code> , <code>noraw</code> , <code>noqiflush</code> , <code>qiflush</code> , <code>timeout</code> , <code>wtimeout</code> , <code>typeahead</code> – curses terminal input option control routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int cbreak(void); int nocbreak(void); int echo(void); int noecho(void); int halfdelay(int <i>tenths</i>); int intrflush(WINDOW *<i>win</i>, bool <i>bf</i>); int keypad(WINDOW *<i>win</i>, bool <i>bf</i>); int meta(WINDOW *<i>win</i>, bool <i>bf</i>); int nodelay(WINDOW *<i>win</i>, bool <i>bf</i>); int notimeout(WINDOW *<i>win</i>, bool <i>bf</i>); int raw(void); int noraw(void); void noqiflush(void); void qiflush(void); void timeout(int <i>delay</i>); void wtimeout(WINDOW *<i>win</i>, int <i>delay</i>); int typeahead(int <i>fildes</i>);</pre> |
| DESCRIPTION | <p>The <code>cbreak()</code> and <code>nocbreak()</code> routines put the terminal into and out of <code>cbreak()</code> mode, respectively. In this mode, characters typed by the user are immediately available to the program, and erase/kill character-processing is not performed. When out of this mode, the tty driver buffers the typed characters until a newline or carriage return is typed. Interrupt and flow control characters are unaffected by this mode. Initially the terminal may or may not be in <code>cbreak()</code> mode, as the mode is inherited; therefore, a program should call <code>cbreak()</code> or <code>nocbreak()</code> explicitly. Most interactive programs using curses set the <code>cbreak()</code> mode.</p> <p>Note that <code>cbreak()</code> overrides <code>raw()</code>. (See <code>curs_getch(3CURSES)</code> for a discussion of how these routines interact with <code>echo()</code> and <code>noecho()</code>.)</p> <p>The <code>echo()</code> and <code>noecho()</code> routines control whether characters typed by the user are echoed by <code>getch()</code> as they are typed. Echoing by the tty driver is always disabled, but initially <code>getch()</code> is in echo mode, so characters typed are echoed. Authors of</p> |

most interactive programs prefer to do their own echoing in a controlled area of the screen, or not to echo at all, so they disable echoing by calling `noecho()`. (See `curs_getch(3CURSES)` for a discussion of how these routines interact with `cbreak()` and `nocbreak()`.)

The `halfdelay()` routine is used for half-delay mode, which is similar to `cbreak()` mode in that characters typed by the user are immediately available to the program. However, after blocking for *tenths* tenths of seconds, `ERR` is returned if nothing has been typed. The value of *tenths* must be a number between 1 and 255. Use `nocbreak()` to leave half-delay mode.

If the `intrflush()` option is enabled, (*bf* is `TRUE`), when an interrupt key is pressed on the keyboard (interrupt, break, quit) all output in the tty driver queue will be flushed, giving the effect of faster response to the interrupt, but causing `curses` to have the wrong idea of what is on the screen. Disabling (*bf* is `FALSE`), the option prevents the flush. The default for the option is inherited from the tty driver settings. The window argument is ignored.

The `keypad()` option enables the keypad of the user's terminal. If enabled (*bf* is `TRUE`), the user can press a function key (such as an arrow key) and `wgetch()` returns a single value representing the function key, as in `KEY_LEFT`. If disabled (*bf* is `FALSE`), `curses` does not treat function keys specially and the program has to interpret the escape sequences itself. If the keypad in the terminal can be turned on (made to transmit) and off (made to work locally), turning on this option causes the terminal keypad to be turned on when `wgetch()` is called. The default value for keypad is false.

Initially, whether the terminal returns 7 or 8 significant bits on input depends on the control mode of the tty driver (see `termio(7I)`). To force 8 bits to be returned, invoke `meta(win, TRUE)`. To force 7 bits to be returned, invoke `meta(win, FALSE)`. The window argument, *win*, is always ignored. If the terminfo capabilities `smm` (`meta_on`) and `rmm` (`meta_off`) are defined for the terminal, `smm` is sent to the terminal when `meta(win, TRUE)` is called and `rmm` is sent when `meta(win, FALSE)` is called.

The `nodelay()` option causes `getch()` to be a non-blocking call. If no input is ready, `getch()` returns `ERR`. If disabled (*bf* is `FALSE`), `getch()` waits until a key is pressed.

While interpreting an input escape sequence, `wgetch()` sets a timer while waiting for the next character. If `notimeout(win, TRUE)` is called, then `wgetch()` does not set a timer. The purpose of the timeout is to differentiate between sequences received from a function key and those typed by a user.

With the `raw()` and `noraw()` routines, the terminal is placed into or out of raw mode. Raw mode is similar to `cbreak()` mode, in that characters typed are immediately passed through to the user program. The differences are that in raw mode, the interrupt, quit, suspend, and flow control characters are all passed through uninterpreted, instead of generating a signal. The behavior of the `BREAK` key depends on other bits in the tty driver that are not set by `curses`.

echo(3CURSES)

When the `noqiflush()` routine is used, normal flush of input and output queues associated with the `INTR`, `QUIT` and `SUSP` characters will not be done (see `termio(7I)`). When `qiflush()` is called, the queues will be flushed when these control characters are read.

The `timeout()` and `wtimeout()` routines set blocking or non-blocking read for a given window. If *delay* is negative, blocking read is used (that is, waits indefinitely for input). If *delay* is zero, then non-blocking read is used (that is, read returns `ERR` if no input is waiting). If *delay* is positive, then read blocks for *delay* milliseconds, and returns `ERR` if there is still no input. Hence, these routines provide the same functionality as `nodelay()`, plus the additional capability of being able to block for only *delay* milliseconds (where *delay* is positive).

`curses` does "line-breakout optimization" by looking for typeahead periodically while updating the screen. If input is found, and it is coming from a tty, the current update is postponed until `refresh()` or `doupdate()` is called again. This allows faster response to commands typed in advance. Normally, the input FILE pointer passed to `newterm()`, or `stdin` in the case that `initscr()` was used, will be used to do this typeahead checking. The `typeahead()` routine specifies that the file descriptor *fildev* is to be used to check for typeahead instead. If *fildev* is `-1`, then no typeahead checking is done.

RETURN VALUES

All routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_getch(3CURSES)`, `curs_initscr(3CURSES)`, `curses(3CURSES)`, `attributes(5)`, `termio(7I)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `echo()`, `noecho()`, `halfdelay()`, `intrflush()`, `meta()`, `nodelay()`, `notimeout()`, `noqiflush()`, `qiflush()`, `timeout()`, and `wtimeout()` may be macros.

| | |
|----------------------|---|
| NAME | echo, noecho – enable/disable terminal echo |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int echo(void); int noecho(void);</pre> |
| DESCRIPTION | The <code>echo()</code> function enables Echo mode for the current screen. The <code>noecho()</code> function disables Echo mode for the current screen. Initially, curses software echo mode is enabled and hardware echo mode of the <code>tty</code> driver is disabled. The <code>echo()</code> and <code>noecho()</code> functions control software echo only. Hardware echo must remain disabled for the duration of the application, else the behavior is undefined. |
| RETURN VALUES | Upon successful completion, these functions return OK. Otherwise, they return ERR. |
| ERRORS | No errors are defined. |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: |

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `getch(3XCURSES)`, `getstr(3XCURSES)`, `initscr(3XCURSES)`, `libcurses(3XCURSES)`, `scanw(3XCURSES)`, `attributes(5)`, `standards(5)`

echochar(3CURSES)

| | |
|----------------------|--|
| NAME | <code>curs_addch</code> , <code>addch</code> , <code>waddch</code> , <code>mvaddch</code> , <code>mvwaddch</code> , <code>echochar</code> , <code>wechochar</code> – add a character (with attributes) to a curses window and advance cursor |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int addch(chtype <i>ch</i>); int waddch(WINDOW *<i>win</i>, chtype <i>ch</i>); int mvaddch(int <i>y</i>, int <i>x</i>, chtype <i>ch</i>); int mvwaddch(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>, chtype <i>ch</i>); int echochar(chtype <i>ch</i>); int wechochar(WINDOW *<i>win</i>, chtype <i>ch</i>);</pre> |
| DESCRIPTION | <p>With the <code>addch()</code>, <code>waddch()</code>, <code>mvaddch()</code>, and <code>mvwaddch()</code> routines, the character <i>ch</i> is put into the window at the current cursor position of the window and the position of the window cursor is advanced. Its function is similar to that of <code>putchar()</code>. At the right margin, an automatic newline is performed. At the bottom of the scrolling region, if <code>scrollok()</code> is enabled, the scrolling region is scrolled up one line.</p> <p>If <i>ch</i> is a tab, newline, or backspace, the cursor is moved appropriately within the window. A newline also does a <code>clrtoeol()</code> before moving. Tabs are considered to be at every eighth column. If <i>ch</i> is another control character, it is drawn in the <code>^X</code> notation. Calling <code>winch()</code> after adding a control character does not return the control character, but instead returns the representation of the control character. See <code>curs_inch(3CURSES)</code>.</p> <p>Video attributes can be combined with a character by OR-ing them into the parameter. This results in these attributes also being set. (The intent here is that text, including attributes, can be copied from one place to another using <code>inch()</code> and <code>addch()</code>.) (see <code>standout()</code>, predefined video attribute constants, on the <code>curs_attr(3CURSES)</code> page).</p> <p>The <code>echochar()</code> and <code>wechochar()</code> routines are functionally equivalent to a call to <code>addch()</code> followed by a call to <code>refresh()</code>, or a call to <code>waddch</code> followed by a call to <code>wrefresh()</code>. The knowledge that only a single character is being output is taken into consideration and, for non-control characters, a considerable performance gain might be seen by using these routines instead of their equivalents.</p> |
| Line Graphics | <p>The following variables may be used to add line drawing characters to the screen with routines of the <code>addch()</code> family. When variables are defined for the terminal, the <code>A_ALTCHARSET</code> bit is turned on (see <code>curs_attr(3CURSES)</code>). Otherwise, the default character listed below is stored in the variable. The names chosen are consistent with the VT100 nomenclature.</p> |

| Name | Default | Glyph Description |
|--------------|---------|-------------------------|
| ACS_ULCORNER | + | upper left-hand corner |
| ACS_LLCORNER | + | lower left-hand corner |
| ACS_URCORNER | + | upper right-hand corner |
| ACS_LRCORNER | + | lower right-hand corner |
| ACS_RTEE | + | right tee |
| ACS_LTEE | + | left tee |
| ACS_BTEE | + | bottom tee |
| ACS_TTEE | + | top tee |
| ACS_HLINE | - | horizontal line |
| ACS_VLINE | | vertical line |
| ACS_PLUS | + | plus |
| ACS_S1 | - | scan line 1 |
| ACS_S9 | - | scan line 9 |
| ACS_DIAMOND | + | diamond |
| ACS_CKBOARD | : | checker board (stipple) |
| ACS_DEGREE | ' | degree symbol |
| ACS_PLMINUS | # | plus/minus |
| ACS_BULLET | o | bullet |
| ACS_LARROW | < | arrow pointing left |
| ACS_RARROW | > | arrow pointing right |
| ACS_DARROW | v | arrow pointing down |
| ACS_UARROW | ^ | arrow pointing up |
| ACS_BOARD | # | board of squares |
| ACS_LANTERN | # | lantern symbol |
| ACS_BLOCK | # | solid square block |

RETURN VALUES

All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

echochar(3CURSES)

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_attr(3CURSES), curs_clear(3CURSES), curs_inch(3CURSES), curs_outopts(3CURSES), curs_refresh(3CURSES), curses(3CURSES), putc(3C), attributes(5)

NOTES The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.

Note that addch(), mvaddch(), mvwaddch(), and echochar() may be macros.

NAME echochar, wechochar – add a single-byte character and refresh window

SYNOPSIS `cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \`
`-R /usr/xpg4/lib -lcurses [library...]`

`c89 [flag...] file... -lcurses [library...]`

```
#include <curses.h>
int echochar(const chtype ch);
int wechochar(WINDOW *win, const chtype ch);
```

DESCRIPTION The echochar() function produces the same effect as calling addch(3XCURSES) and then refresh(3XCURSES). The wechochar() function produces the same effect as calling waddch(3XCURSES) and then wrefresh(3XCURSES).

PARAMETERS *ch* Is a pointer to the character to be written to the window.
win Is a pointer to the window in which the character is to be added.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO addch(3XCURSES), doupdate(3XCURSES), echo_wchar(3XCURSES),
 libcurses(3XCURSES), attributes(5), standards(5)

echowchar(3CURSES)

| | |
|----------------------|--|
| NAME | <code> curs_addwch, addwch, waddwch, mvaddwch, mvwaddwch, echowchar, wechowchar</code> – add a <code>wchar_t</code> character (with attributes) to a curses window and advance cursor |
| SYNOPSIS | <pre>cc [flag...] file... -lcurses [library...] #include<curses.h> int addwch(chtype <i>wch</i>); int waddwch(WINDOW *<i>win</i>, chtype <i>wch</i>); int mvaddwch(int <i>y</i>, int <i>x</i>, chtype <i>wch</i>); int mvwaddwch(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>, chtype <i>wch</i>); int echowchar(chtype <i>wch</i>); int wechowchar(WINDOW *<i>win</i>, chtype <i>wch</i>);</pre> |
| DESCRIPTION | <p>The <code>addwch()</code>, <code>waddwch()</code>, <code>mvaddwch()</code>, and <code>mvwaddwch()</code> routines put the character <code>wch</code>, holding a <code>wchar_t</code> character, into the window at the current cursor position of the window and advance the position of the window cursor. Their function is similar to that of <code>putwchar(3C)</code> in the C multibyte library. At the right margin, an automatic newline is performed. At the bottom of the scrolling region, if <code>scrollok</code> is enabled, the scrolling region is scrolled up one line.</p> <p>If <code>wch</code> is a tab, newline, or backspace, the cursor is moved appropriately within the window. A newline also does a <code>clrtoeol(3CURSES)</code> before moving. Tabs are considered to be at every eighth column. If <code>wch</code> is another control character, it is drawn in the <code>^X</code> notation. Calling <code>winwch(3CURSES)</code> after adding a control character does not return the control character, but instead returns the representation of the control character.</p> <p>Video attributes can be combined with a <code>wchar_t</code> character by OR-ing them into the parameter. This results in these attributes also being set. (The intent here is that text, including attributes, can be copied from one place to another using <code>inwch()</code> and <code>addwch()</code>.) See <code>standout(3CURSES)</code>, predefined video attribute constants.</p> <p>The <code>echowchar()</code> and <code>wechowchar()</code> routines are functionally equivalent to a call to <code>addwch()</code> followed by a call to <code>refresh(3CURSES)</code>, or a call to <code>waddwch()</code> followed by a call to <code>wrefresh(3CURSES)</code>. The knowledge that only a single character is being output is taken into consideration and, for non-control characters, a considerable performance gain might be seen by using these routines instead of their equivalents.</p> |
| Line Graphics | <p>The following variables may be used to add line drawing characters to the screen with routines of the <code>addwch()</code> family. When variables are defined for the terminal, the <code>A_ALTCHARSET</code> bit is turned on. (See <code> curs_attr(3CURSES)</code>). Otherwise, the default character listed below is stored in the variable. The names chosen are consistent with the VT100 nomenclature.</p> |

echowchar(3CURSES)

| Name | Default | Glyph Description |
|--------------|---------|-------------------------|
| ACS_ULCORNER | + | upper left-hand corner |
| ACS_LLCORNER | + | lower left-hand corner |
| ACS_URCORNER | + | upper right-hand corner |
| ACS_LRCORNER | + | lower right-hand corner |
| ACS_RTEE | + | right tee |
| ACS_LTEE | + | left tee |
| ACS_BTEE | + | bottom tee |
| ACS_TTEE | + | top tee |
| ACS_HLINE | - | horizontal line |
| ACS_VLINE | | vertical line |
| ACS_PLUS | + | plus |
| ACS_S1 | - | scan line 1 |
| ACS_S9 | - | scan line 9 |
| ACS_DIAMOND | + | diamond |
| ACS_CKBOARD | : | checker board (stipple) |
| ACS_DEGREE | ' | degree symbol |
| ACS_PLMINUS | # | plus/minus |
| ACS_BULLET | o | bullet |
| ACS_LARROW | < | arrow pointing left |
| ACS_RARRROW | > | arrow pointing right |
| ACS_DARROW | v | arrow pointing down |
| ACS_UARROW | ^ | arrow pointing up |
| ACS_BOARD | # | board of squares |
| ACS_LANTERN | # | lantern symbol |
| ACS_BLOCK | # | solid square block |

RETURN VALUE

All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion, unless otherwise noted in the preceding routine descriptions.

echowchar(3CURSES)

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO putwchar(3C), clrtoeol(3CURSES), curses(3CURSES), curs_attr(3CURSES), curs_inwch(3CURSES), curs_outopts(3CURSES), refresh(3CURSES), standout(3CURSES), winch(3CURSES), wrefresh(3CURSES), attributes(5)

NOTES The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <widec.h>.

Note that addwch(), mvaddwch(), mvwaddwch(), and echowchar() may be macros.

None of these routines can use the color attribute in chtype.

NAME echo_wchar, wecho_wchar – add a complex character and refresh window

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

int echo_wchar(const cchar_t *wch);

int wecho_wchar(WINDOW *win, const cchar_t *wch);
```

DESCRIPTION The echo_wchar() function produces the same effect as calling add_wch(3XCURSES) and then refresh(3XCURSES). The wecho_wchar() function produces the same effect as calling wadd_wch(3XCURSES) and then wrefresh(3XCURSES).

PARAMETERS

wch Is a pointer to the complex character to be written to the window.

win Is a pointer to the window in which the character is to be added.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO add_wch(3XCURSES), doupdate(3XCURSES), echochar(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

endwin(3CURSES)

| | |
|--------------------|--|
| NAME | <code>curs_initscr</code> , <code>initscr</code> , <code>newterm</code> , <code>endwin</code> , <code>isendwin</code> , <code>set_term</code> , <code>delscreen</code> – curses screen initialization and manipulation routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> WINDOW *initscr(void); int endwin(void); int isendwin(void); SCREEN *newterm(char *<i>type</i>, FILE *<i>outfd</i>, FILE *<i>infd</i>); SCREEN *set_term(SCREEN *<i>new</i>); void delscreen(SCREEN * <i>sp</i>);</pre> |
| DESCRIPTION | <p><code>initscr()</code> is almost always the first routine that should be called (the exceptions are <code>slk_init()</code>, <code>filter()</code>, <code>ripoffline()</code>, <code>use_env()</code> and, for multiple-terminal applications, <code>newterm()</code>.) This determines the terminal type and initializes all curses data structures. <code>initscr()</code> also causes the first call to <code>refresh()</code> to clear the screen. If errors occur, <code>initscr()</code> writes an appropriate error message to standard error and exits; otherwise, a pointer is returned to <code>stdscr()</code>. If the program needs an indication of error conditions, <code>newterm()</code> should be used instead of <code>initscr()</code>; <code>initscr()</code> should only be called once per application.</p> <p>A program that outputs to more than one terminal should use the <code>newterm()</code> routine for each terminal instead of <code>initscr()</code>. A program that needs an indication of error conditions, so it can continue to run in a line-oriented mode if the terminal cannot support a screen-oriented program, would also use this routine. The routine <code>newterm()</code> should be called once for each terminal. It returns a variable of type <code>SCREEN *</code> which should be saved as a reference to that terminal. The arguments are the <i>type</i> of the terminal to be used in place of <code>\$TERM</code>, a file pointer for output to the terminal, and another file pointer for input from the terminal (if <i>type</i> is <code>NULL</code>, <code>\$TERM</code> will be used). The program must also call <code>endwin()</code> for each terminal being used before exiting from curses. If <code>newterm()</code> is called more than once for the same terminal, the first terminal referred to must be the last one for which <code>endwin()</code> is called.</p> <p>A program should always call <code>endwin()</code> before exiting or escaping from curses mode temporarily. This routine restores tty modes, moves the cursor to the lower left-hand corner of the screen and resets the terminal into the proper non-visual mode. Calling <code>refresh()</code> or <code>doupdate()</code> after a temporary escape causes the program to resume visual mode.</p> <p>The <code>isendwin()</code> routine returns <code>TRUE</code> if <code>endwin()</code> has been called without any subsequent calls to <code>wrefresh()</code>, and <code>FALSE</code> otherwise.</p> |

endwin(3CURSES)

The `set_term()` routine is used to switch between different terminals. The screen reference `new` becomes the new current terminal. The previous terminal is returned by the routine. This is the only routine which manipulates `SCREEN` pointers; all other routines affect only the current terminal.

The `delscreen()` routine frees storage associated with the `SCREEN` data structure. The `endwin()` routine does not do this, so `delscreen()` should be called after `endwin()` if a particular `SCREEN` is no longer needed.

RETURN VALUES `endwin()` returns the integer `ERR` upon failure and `OK` upon successful completion.

Routines that return pointers always return `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_kernel(3CURSES)`, `curl_refresh(3CURSES)`, `curl_slk(3CURSES)`, `curl_util(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `initscr()` and `newterm()` may be macros.

endwin(3XCURSES)

| NAME | endwin, isendwin – restore initial terminal environment | | | | | | |
|----------------------|--|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int endwin(void); bool isendwin(void);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>endwin()</code> function restores the terminal after Curses activity by at least restoring the saved shell terminsl mode, flushing any output to the terminal, and moving the cursor to the first column of the last line of the screen. Refreshing a window resumes program mode. The application must call <code>endwin()</code> for each terminal being used before exiting. If <code>newterm(3XCURSES)</code> is called more than once for the same terminal, the first screen created must be the last one for which <code>endwin()</code> is called.</p> <p>The <code>isendwin()</code> function indicates whether or not a screen has been refreshed since the last call to <code>endwin()</code>.</p> | | | | | | |
| RETURN VALUES | <p>Upon successful completion, the <code>endwin()</code> function returns OK. Otherwise, it returns ERR.</p> <p>The <code>isendwin()</code> function returns TRUE if <code>endwin()</code> has been called without any subsequent refresh. Otherwise, it returns FALSE.</p> | | | | | | |
| ERRORS | Non errors are defined. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Standard</td></tr><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>douppdate(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>newterm(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

| NAME | curs_clear, erase, werase, clear, wclear, clrrobot, wclrrobot, clrtoeol, wclrtoeol – clear all or part of a curses window | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file ... -lcurses [library ...] #include <curses.h> int erase(void); int werase(WINDOW *win); int clear(void); int wclear(WINDOW *win); int clrrobot(void); int wclrrobot(WINDOW *win); int clrtoeol(void); int wclrtoeol(WINDOW *win);</pre> | | | | |
| DESCRIPTION | <p>The erase() and werase() routines copy blanks to every position in the window.</p> <p>The clear() and wclear() routines are like erase() and werase(), but they also call clearok(), so that the screen is cleared completely on the next call to wrefresh() for that window and repainted from scratch.</p> <p>The clrrobot() and wclrrobot() routines erase all lines below the cursor in the window. Also, the current line to the right of the cursor, inclusive, is erased.</p> <p>The clrtoeol() and wclrtoeol() routines erase the current line to the right of the cursor, inclusive.</p> | | | | |
| RETURN VALUES | All routines return the integer OK, or a non-negative integer if immedok() is set. See curs_ouptops(3CURSES). | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curs_ouptops(3CURSES), curs_refresh(3CURSES), curses(3CURSES), attributes(5) | | | | |
| NOTES | <p>The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.</p> <p>Note that erase(), werase(), clear(), wclear(), clrrobot(), and clrtoeol() may be macros.</p> | | | | |

erase(3PLOT)

| | |
|--------------------|--|
| NAME | plot, arc, box, circle, closepl, closevt, cont, erase, label, line, linemod, move, openpl, openvt, point, space – graphics interface |
| SYNOPSIS | <pre>cc [flag ...] file ... -lplot [library...] #include <plot.h> void arc(short x0, short y0, short x1, short y1, short x2, short y2); void box(short x0, short y0, short x1, short y1); void circle(short x, short y, short r); void closepl(); void closevt(); void cont(short x, short y); void erase(); void label(char *s); void line(short x0, short y0, short x1, short y1); void linemod(char *s); void move(short x, short y); void openpl(); void openvt(); void point(short x, short y); void space(short x0, short y0, short x1, short y1);</pre> |
| DESCRIPTION | <p>These functions generate graphics output for a set of output devices. The format of the output is dependent upon which link editor option is used when the program is compiled and linked (see Link Editor).</p> <p>The term "current point" refers to the current setting for the x and y coordinates.</p> <p>The <code>arc()</code> function specifies a circular arc. The coordinates $(x0, y0)$ specify the center of the arc. The coordinates $(x1, y1)$ specify the starting point of the arc. The coordinates $(x2, y2)$ specify the end point of the circular arc.</p> <p>The <code>box()</code> function specifies a rectangle with coordinates $(x0, y0)$, $(x0, y1)$, $(x1, y0)$, and $(x1, y1)$. The current point is set to $(x1, y1)$.</p> <p>The <code>circle()</code> function specifies a circle with a center at the coordinates (x, y) and a radius of r.</p> <p>The <code>closevt()</code> and <code>closepl()</code> functions flush the output.</p> <p>The <code>cont()</code> function specifies a line beginning at the current point and ending at the coordinates (x, y). The current point is set to (x, y).</p> |

The `erase()` function starts another frame of output.

The `label()` function places the null terminated string `s` so that the first character falls on the current point. The string is then terminated by a NEWLINE character.

The `line()` function draws a line starting at the coordinates $(x0, y0)$ and ending at the coordinates $(x1, y1)$. The current point is set to $(x1, y1)$.

The `linemod()` function specifies the style for drawing future lines. `s` may contain one of the following: `dotted`, `solid`, `longdashed`, `shortdashed`, or `dotdashed`.

The `move()` function sets the current point to the coordinates (x, y) .

The `openpl()` or `openvt()` function must be called to open the device before any other plot functions are called.

The `point()` function plots the point given by the coordinates (x, y) . The current point is set to (x, y) .

The `space()` function specifies the size of the plotting area. The plot will be reduced or enlarged as necessary to fit the area specified. The coordinates $(x0, y0)$ specify the lower left hand corner of the plotting area. The coordinates $(x1, y1)$ specify the upper right hand corner of the plotting area.

Link Editor Various flavors of these functions exist for different output devices. They are obtained by using the following `ld(1)` options:

```
-lplot          device-independent graphics stream on standard output in the
                 format described in plot(4B)
-l300           GSI 300 terminal
-l300s          GSI 300S terminal
-l4014          Tektronix 4014 terminal
-l450           GSI 450 terminal
-lvt0
```

FILES

```
/usr/lib/libplot.a
  archive library
/usr/lib/libplot.so.1
  shared object
/usr/lib/sparcv9/libplot.so.1
  64-bit shared object
/usr/lib/lib300.a
  archive library
/usr/lib/lib300.so.1
  shared object
```

erase(3PLOT)

/usr/lib/sparcv9/lib300.so.1
64-bit shared object

/usr/lib/lib300s.a
archive library

/usr/lib/lib300s.so.1
shared object

/usr/lib/sparcv9/lib300s.so.1
64-bit shared object

/usr/lib/lib4014.a
archive library

/usr/lib/lib4014.so.1
shared object

/usr/lib/sparcv9/lib4014.so.1
64-bit shared object

/usr/lib/lib450.a
archive library

/usr/lib/lib450.so.1
shared object

/usr/lib/sparcv9/lib450.so.1
64-bit shared object

/usr/lib/libvt0.a
archive library

/usr/lib/libvt0.so.1
shared object

/usr/lib/sparcv9/libvt0.so.1
64-bit shared object

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `graph(1)`, `ld(1)`, `libplot(3LIB)`, `plot(4B)`, `attributes(5)`

| NAME | clear, erase, wclear, werase – clear a window | | | | | | |
|---------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int clear(void); int erase(void); int wclear(WINDOW *win); int werase(WINDOW *win);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>clear()</code> and <code>erase()</code> functions clear <code>stdscr</code>, destroying its previous contents. The <code>wclear()</code> and <code>werase()</code> functions perform the same action, but clear the window specified by <code>win</code> instead of <code>stdscr</code>.</p> <p>The <code>clear()</code> and <code>wclear()</code> functions also call the <code>clearok()</code> function. This function clears and redraws the entire screen on the next call to <code>refresh(3XCURSES)</code> or <code>wrefresh(3XCURSES)</code> for the window.</p> <p>The current background character (and attributes) is used to clear the screen.</p> | | | | | | |
| PARAMETERS | <p><i>win</i> Is a pointer to the window that is to be cleared.</p> | | | | | | |
| ERRORS | <p>OK Successful completion.</p> <p>ERR An error occurred.</p> | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <p><code>bkgdset(3XCURSES)</code>, <code>clearok(3XCURSES)</code>, <code>clrtoeol(3XCURSES)</code>, <code>clrtoeol(3XCURSES)</code>, <code>doupdate(3XCURSES)</code>, <code>libcurses(3XCURSES)</code>, <code>refresh(3XCURSES)</code>, <code>wrefresh(3XCURSES)</code>, <code>attributes(5)</code>, <code>standards(5)</code></p> | | | | | | |

erasechar(3CURSES)

| | |
|----------------------|---|
| NAME | <code>curs_termattrs</code> , <code>baudrate</code> , <code>erasechar</code> , <code>has_ic</code> , <code>has_il</code> , <code>killchar</code> , <code>longname</code> , <code>termattrs</code> , <code>termname</code> – curses environment query routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int baudrate(void); char erasechar(void); int has_ic(void); int has_il(void); char killchar(void); char *longname(void); chtype termattrs(void); char *termname(void);</pre> |
| DESCRIPTION | <p>The <code>baudrate()</code> routine returns the output speed of the terminal. The number returned is in bits per second, for example 9600, and is an integer.</p> <p>With the <code>erasechar()</code> routine, the user's current erase character is returned.</p> <p>The <code>has_ic()</code> routine is true if the terminal has insert- and delete-character capabilities.</p> <p>The <code>has_il()</code> routine is true if the terminal has insert- and delete-line capabilities, or can simulate them using scrolling regions. This might be used to determine if it would be appropriate to turn on physical scrolling using <code>scrollok()</code>.</p> <p>With the <code>killchar()</code> routine, the user's current line kill character is returned.</p> <p>The <code>longname()</code> routine returns a pointer to a static area containing a verbose description of the current terminal. The maximum length of a verbose description is 128 characters. It is defined only after the call to <code>initscr()</code> or <code>newterm()</code>. The area is overwritten by each call to <code>newterm()</code> and is not restored by <code>set_term()</code>, so the value should be saved between calls to <code>newterm()</code> if <code>longname()</code> is going to be used with multiple terminals.</p> <p>If a given terminal doesn't support a video attribute that an application program is trying to use, <code>curses</code> may substitute a different video attribute for it. The <code>termattrs()</code> function returns a logical OR of all video attributes supported by the terminal. This information is useful when a <code>curses</code> program needs complete control over the appearance of the screen.</p> <p>The <code>termname()</code> routine returns the value of the environment variable <code>TERM</code> (truncated to 14 characters).</p> |
| RETURN VALUES | <code>longname()</code> and <code>termname()</code> return <code>NULL</code> on error. |

erasechar(3CURSES)

Routines that return an integer return ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_initscr(3CURSES), curs_outopts(3CURSES), curses(3CURSES), attributes(5)

NOTES The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.

Note that termattrs() may be a macro.

erasechar(3XCURSES)

| NAME | erasechar, eraseswchar, killchar, killwchar – return current ERASE or KILL characters | | | | | | |
|----------------------|--|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> char erasechar(void); int eraseswchar(wchar_t *ch); char killchar(void); int killwchar(wchar_t *ch);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>erasechar()</code> function returns the current ERASE character from the tty driver. This character is used to delete the previous character during keyboard input. The returned value can be used when including deletion capability in interactive programs.</p> <p>The <code>killchar()</code> function is similar to <code>erasechar()</code>. It returns the current KILL character.</p> <p>The <code>eraseswchar()</code> and <code>killwchar()</code> functions are similar to <code>erasechar()</code> and <code>killchar()</code> respectively, but store the ERASE or KILL character in the object pointed to by <i>ch</i>.</p> | | | | | | |
| PARAMETERS | <p><i>ch</i> Is a pointer to a location where a character may be stored.</p> | | | | | | |
| RETURN VALUES | <p>For <code>erasechar()</code> and <code>killchar()</code>, the terminal's current ERASE or KILL character is returned.</p> <p>On success, the <code>eraseswchar()</code> and <code>killwchar()</code> functions return OK. Otherwise, they return ERR.</p> | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Standard</td></tr><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>getch(3XCURSES)</code> , <code>getstr(3XCURSES)</code> , <code>get_wch(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

| NAME | erasechar, eraseswchar, killchar, killwchar – return current ERASE or KILL characters | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> char erasechar(void); int eraseswchar(wchar_t *ch); char killchar(void); int killwchar(wchar_t *ch);</pre> | | | | | | |
| DESCRIPTION | <p>The erasechar() function returns the current ERASE character from the tty driver. This character is used to delete the previous character during keyboard input. The returned value can be used when including deletion capability in interactive programs.</p> <p>The killchar() function is similar to erasechar(). It returns the current KILL character.</p> <p>The eraseswchar() and killwchar() functions are similar to erasechar() and killchar() respectively, but store the ERASE or KILL character in the object pointed to by <i>ch</i>.</p> | | | | | | |
| PARAMETERS | <i>ch</i> Is a pointer to a location where a character may be stored. | | | | | | |
| RETURN VALUES | <p>For erasechar() and killchar(), the terminal's current ERASE or KILL character is returned.</p> <p>On success, the eraseswchar() and killwchar() functions return OK. Otherwise, they return ERR.</p> | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | getch(3XCURSES), getstr(3XCURSES), get_wch(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5) | | | | | | |

field_arg(3CURSES)

NAME form_field_validation, set_field_type, field_type, field_arg – forms field data type validation

SYNOPSIS

```
cc [ flag... ] file... -lform -lcurses [ library... ]
#include <form.h>

int set_field_type(FIELD *field, FIELDTYPE *type, ...);
FIELDTYPE *field_type(FIELD *field);
char *field_arg(FIELD *field);
```

DESCRIPTION set_field_type() associates the specified field type with *field*. Certain field types take additional arguments. TYPE_ALNUM, for instance, requires one, the minimum width specification for the field. The other predefined field types are: TYPE_ALPHA, TYPE_ENUM, TYPE_INTEGER, TYPE_NUMERIC, and TYPE_REGEXP.

field_type() returns a pointer to the field type of *field*. NULL is returned if no field type is assigned.

field_arg() returns a pointer to the field arguments associated with the field type of *field*. NULL is returned if no field type is assigned.

RETURN VALUES field_type() and field_arg() return NULL on error.

set_field_type() returns one of the following:

| | |
|----------------|-------------------------------------|
| E_OK | The function returned successfully. |
| E_SYSTEM_ERROR | System error. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), forms(3CURSES), attributes(5)

NOTES The header <form.h> automatically includes the headers <eti.h> and <curses.h>.

| NAME | form_field_attributes, set_field_fore, field_fore, set_field_back, field_back, set_field_pad, field_pad – format the general display attributes of forms | | | | | | |
|----------------------|--|----------------|-------------------------------------|----------------|---------------|----------------|---------------------------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_field_fore(FIELD *field, chtype attr); chtype field_fore(FIELD *field); int set_field_back(FIELD *field, chtype attr); chtype field_back(FIELD *field); int set_field_pad(FIELD *field, int pad); int field_pad(FIELD *field);</pre> | | | | | | |
| DESCRIPTION | <p>set_field_fore() sets the foreground attribute of <i>field</i>. The foreground attribute is the low-level curses display attribute used to display the field contents. field_fore() returns the foreground attribute of <i>field</i>.</p> <p>set_field_back() sets the background attribute of <i>field</i>. The background attribute is the low-level curses display attribute used to display the extent of the field. field_back() returns the background attribute of <i>field</i>.</p> <p>set_field_pad() sets the pad character of <i>field</i> to <i>pad</i>. The pad character is the character used to fill within the field. field_pad() returns the pad character of <i>field</i>.</p> | | | | | | |
| RETURN VALUES | <p>field_fore(), field_back(), and field_pad() return default values if <i>field</i> is NULL. If <i>field</i> is not NULL and is not a valid FIELD pointer, the return value from these routines is undefined.</p> <p>set_field_fore(), set_field_back(), and set_field_pad() return one of the following:</p> <table border="0"> <tr> <td style="padding-right: 20px;">E_OK</td> <td>The function returned successfully.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> <tr> <td>E_BAD_ARGUMENT</td> <td>An argument is incorrect.</td> </tr> </table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An argument is incorrect. |
| E_OK | The function returned successfully. | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | |
| E_BAD_ARGUMENT | An argument is incorrect. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | | | |

field_buffer(3CURSES)

NAME form_field_buffer, set_field_buffer, field_buffer, set_field_status, field_status, set_max_field – set and get forms field attributes

SYNOPSIS

```
cc [ flag... ] file... -lform -lcurses [ library... ]
#include <form.h>

int set_field_buffer(FIELD *field, int buf, char *value);
char *field_buffer(FIELD *field, int buf);
int set_field_status(FIELD *field, int status);
int field_status(FIELD *field);
int set_max_field(FIELD *field, int max);
```

DESCRIPTION set_field_buffer() sets buffer *buf* of *field* to *value*. Buffer 0 stores the displayed contents of the field. Buffers other than 0 are application specific and not used by the forms library routines. field_buffer() returns the value of *field* buffer *buf*.

Every field has an associated status flag that is set whenever the contents of field buffer 0 changes. set_field_status() sets the status flag of *field* to *status*. field_status() returns the status of *field*.

set_max_field() sets a maximum growth on a dynamic field, or if *max*=0 turns off any maximum growth.

RETURN VALUES field_buffer() returns NULL on error.

field_status() returns TRUE or FALSE.

set_field_buffer(), set_field_status(), and set_max_field() return one of the following:

| | |
|----------------|-------------------------------------|
| E_OK | The function returned successfully. |
| E_SYSTEM_ERROR | System error |
| E_BAD_ARGUMENT | An argument is incorrect. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), forms(3CURSES), attributes(5)

NOTES The header <form.h> automatically includes the headers <eti.h> and <curses.h>.

| NAME | form_field, set_form_fields, form_fields, field_count, move_field – connect fields to forms | | | | | | | | | | |
|----------------------|--|----------------|-------------------------------------|-------------|---|----------------|---------------|----------------|--------------------------|----------|---------------------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_form_fields(FORM *form, FIELD **field); FIELD **form_fields(FORM *form); int field_count(FORM *form); int move_field(FIELD *field, int frow, int fcol);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>set_form_fields() changes the fields connected to <i>form</i> to <i>fields</i>. The original fields are disconnected.</p> <p>form_fields() returns a pointer to the field pointer array connected to <i>form</i>.</p> <p>field_count() returns the number of fields connected to <i>form</i>.</p> <p>move_field() moves the disconnected <i>field</i> to the location <i>frow</i>, <i>fcol</i> in the forms subwindow.</p> | | | | | | | | | | |
| RETURN VALUES | <p>form_fields() returns NULL on error.</p> <p>field_count() returns -1 on error.</p> <p>set_form_fields() and move_field() return one of the following:</p> <table border="0"> <tr> <td>E_OK</td> <td>The function returned successfully.</td> </tr> <tr> <td>E_CONNECTED</td> <td>The field is already connected to a form.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> <tr> <td>E_BAD_ARGUMENT</td> <td>An argument is incorrect</td> </tr> <tr> <td>E_POSTED</td> <td>The form is posted.</td> </tr> </table> | E_OK | The function returned successfully. | E_CONNECTED | The field is already connected to a form. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An argument is incorrect | E_POSTED | The form is posted. |
| E_OK | The function returned successfully. | | | | | | | | | | |
| E_CONNECTED | The field is already connected to a form. | | | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | | | |
| E_BAD_ARGUMENT | An argument is incorrect | | | | | | | | | | |
| E_POSTED | The form is posted. | | | | | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | | | | | |
| | <table border="1"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | | | | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | | | | | |
| MT-Level | Unsafe | | | | | | | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | | | | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | | | | | | | |

field_fore(3CURSES)

| NAME | form_field_attributes, set_field_fore, field_fore, set_field_back, field_back, set_field_pad, field_pad – format the general display attributes of forms | | | | | | |
|----------------------|--|----------------|-------------------------------------|----------------|---------------|----------------|---------------------------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_field_fore(FIELD *field, chtype attr); chtype field_fore(FIELD *field); int set_field_back(FIELD *field, chtype attr); chtype field_back(FIELD *field); int set_field_pad(FIELD *field, int pad); int field_pad(FIELD *field);</pre> | | | | | | |
| DESCRIPTION | <p>set_field_fore() sets the foreground attribute of <i>field</i>. The foreground attribute is the low-level curses display attribute used to display the field contents. field_fore() returns the foreground attribute of <i>field</i>.</p> <p>set_field_back() sets the background attribute of <i>field</i>. The background attribute is the low-level curses display attribute used to display the extent of the field. field_back() returns the background attribute of <i>field</i>.</p> <p>set_field_pad() sets the pad character of <i>field</i> to <i>pad</i>. The pad character is the character used to fill within the field. field_pad() returns the pad character of <i>field</i>.</p> | | | | | | |
| RETURN VALUES | <p>field_fore(), field_back(), and field_pad() return default values if <i>field</i> is NULL. If <i>field</i> is not NULL and is not a valid FIELD pointer, the return value from these routines is undefined.</p> <p>set_field_fore(), set_field_back(), and set_field_pad() return one of the following:</p> <table><tr><td>E_OK</td><td>The function returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr><tr><td>E_BAD_ARGUMENT</td><td>An argument is incorrect.</td></tr></table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An argument is incorrect. |
| E_OK | The function returned successfully. | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | |
| E_BAD_ARGUMENT | An argument is incorrect. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | | | |

NAME form_page, set_form_page, set_current_field, current_field, field_index – set forms current page and field

SYNOPSIS

```
cc [ flag... ] file... -lform -lcurses [ library... ]
#include <form.h>

int set_form_page(FORM *form, int page);
int form_page(FORM *form);
int set_current_field(FORM *form, FIELD *field);
FIELD *current_field(FORM*form);
int field_index(FIELD *field);
```

DESCRIPTION

set_form_page() sets the page number of *form* to *page*. form_page() returns the current page number of *form*.

set_current_field() sets the current field of *form* to *field*. current_field() returns a pointer to the current field of *form*.

field_index() returns the index in the field pointer array of *field*.

RETURN VALUES

form_page() returns -1 on error.

current_field() returns NULL on error.

field_index() returns -1 on error.

set_form_page() and set_current_field() return one of the following:

| | |
|------------------|--|
| E_OK | The function returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An argument is incorrect. |
| E_BAD_STATE | The routine was called from an initialization or termination function. |
| E_INVALID_FIELD | The field contents are invalid. |
| E_REQUEST_DENIED | The form driver request failed |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), forms(3CURSES), attributes(5)

field_index(3CURSES)

NOTES | The header `<form.h>` automatically includes the headers `<eti.h>` and `<curses.h>`.

| NAME | form_field_info, field_info, dynamic_field_info – get forms field characteristics | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int field_info(FIELD *field, int *rows, int *cols, int *frow, int *fcol, int *nrow, int *nbuf); int dynamic_field_info(FIELD *field, int *drows, int *dcols, int *max);</pre> | | | | |
| DESCRIPTION | <p>field_info() returns the size, position, and other named field characteristics, as defined in the original call to new_field(), to the locations pointed to by the arguments rows, cols, frow, fcol, nrow, and nbuf.</p> <p>dynamic_field_info() returns the actual size of the field in the pointer arguments drows, dcols and returns the maximum growth allowed for field in max. If no maximum growth limit is specified for field, max will contain 0. A field can be made dynamic by turning off the field option O_STATIC.</p> | | | | |
| RETURN VALUES | <p>These routines return one of the following:</p> <p>E_OK The function returned successfully.</p> <p>E_SYSTEM_ERROR System error.</p> <p>E_BAD_ARGUMENT An argument is incorrect.</p> | | | | |
| ATTRIBUTES | <p>See attributes(5) for descriptions of the following attributes:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | |

field_init(3CURSES)

| | | | | | |
|-----------------------------|--|-------------------|-------------------------------------|-----------------------------|---------------|
| NAME | form_hook, set_form_init, form_init, set_form_term, form_term, set_field_init, field_init, set_field_term, field_term – assign application-specific routines for invocation by forms | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_form_init(FORM *form, void (*func)(FORM*)); void (*form_init)(FORM *form); int set_form_term(FORM *form, void (*func)(FORM*)); void (*form_term)(FORM *form); int set_field_init(FORM *form, void (*func)(FORM*)); void (*field_init)(FORM *form); int set_field_term(FORM *form, void (*func)(FORM*)); void (*field_term)(FORM *form);</pre> | | | | |
| DESCRIPTION | <p>These routines allow the programmer to assign application specific routines to be executed automatically at initialization and termination points in the <code>forms</code> application. The user need not specify any application-defined initialization or termination routines at all, but they may be helpful for displaying messages or page numbers and other chores.</p> <p><code>set_form_init()</code> assigns an application-defined initialization function to be called when the <code>form</code> is posted and just after a page change. <code>form_init()</code> returns a pointer to the initialization function, if any.</p> <p><code>set_form_term()</code> assigns an application-defined function to be called when the <code>form</code> is unposted and just before a page change. <code>form_term()</code> returns a pointer to the function, if any.</p> <p><code>set_field_init()</code> assigns an application-defined function to be called when the <code>form</code> is posted and just after the current field changes. <code>field_init()</code> returns a pointer to the function, if any.</p> <p><code>set_field_term()</code> assigns an application-defined function to be called when the <code>form</code> is unposted and just before the current field changes. <code>field_term()</code> returns a pointer to the function, if any.</p> | | | | |
| RETURN VALUES | <p>Routines that return pointers always return <code>NULL</code> on error. Routines that return an integer return one of the following:</p> <table><tr><td><code>E_OK</code></td><td>The function returned successfully.</td></tr><tr><td><code>E_SYSTEM_ERROR</code></td><td>System error.</td></tr></table> | <code>E_OK</code> | The function returned successfully. | <code>E_SYSTEM_ERROR</code> | System error. |
| <code>E_OK</code> | The function returned successfully. | | | | |
| <code>E_SYSTEM_ERROR</code> | System error. | | | | |

field_init(3CURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3CURSES)`, `forms(3CURSES)`, `attributes(5)`

NOTES The header `<form.h>` automatically includes the headers `<eti.h>` and `< curses.h>`.

field_just(3CURSES)

NAME | form_field_just, set_field_just, field_just – format the general appearance of forms

SYNOPSIS |

```
cc [ flag... ] file... -lform -lcurses [ library... ]
#include <form.h>

int set_field_just(FIELD *field, int justification);
int field_just(FIELD *field);
```

DESCRIPTION | set_field_just() sets the justification for *field*. Justification may be one of:

NO_JUSTIFICATION
JUSTIFY_RIGHT
JUSTIFY_LEFT
JUSTIFY_CENTER

.

The field justification will be ignored if *field* is a dynamic field.

field_just() returns the type of justification assigned to *field*.

RETURN VALUES | field_just() returns one of the following:

NO_JUSTIFICATION
JUSTIFY_RIGHT
JUSTIFY_LEFT
JUSTIFY_CENTER.

set_field_just() returns one of the following:

E_OK The function returned successfully.
E_SYSTEM_ERROR System error.
E_BAD_ARGUMENT An argument is incorrect.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5)

NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>.

| | | | | | | | | | | | | | | | | | | | | | |
|----------------------|--|-----------|-------------------------------------|----------------|---|-----------|--|--------|--------------------------|--------|---|---------|---|------------|---|----------|------------------------------------|----------|--------------------------------------|----------|--|
| NAME | form_field_opts, set_field_opts, field_opts_on, field_opts_off, field_opts – forms field option routines | | | | | | | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_field_opts(FIELD *field, OPTIONS opts); int set_field_opts(FIELD *field, OPTIONS opts); int field_opts_on(FIELD *field, OPTIONS opts); int field_opts_off(FIELD *field, OPTIONS opts); OPTIONS field_opts(FIELD *field);</pre> | | | | | | | | | | | | | | | | | | | | |
| DESCRIPTION | <p>set_field_opts() turns on the named options of <i>field</i> and turns off all remaining options. Options are boolean values that can be OR-ed together.</p> <p>field_opts_on() turns on the named options; no other options are changed.</p> <p>field_opts_off() turns off the named options; no other options are changed.</p> <p>field_opts() returns the options set for <i>field</i>.</p> <table border="0"> <tr> <td style="padding-right: 20px;">O_VISIBLE</td> <td>The field is displayed.</td> </tr> <tr> <td>O_ACTIVE</td> <td>The field is visited during processing.</td> </tr> <tr> <td>O_PUBLIC</td> <td>The field contents are displayed as data is entered.</td> </tr> <tr> <td>O_EDIT</td> <td>The field can be edited.</td> </tr> <tr> <td>O_WRAP</td> <td>Words not fitting on a line are wrapped to the next line.</td> </tr> <tr> <td>O_BLANK</td> <td>The whole field is cleared if a character is entered in the first position.</td> </tr> <tr> <td>O_AUTOSKIP</td> <td>Skip to the next field when the current field becomes full.</td> </tr> <tr> <td>O_NULLOK</td> <td>A blank field is considered valid.</td> </tr> <tr> <td>O_STATIC</td> <td>The field buffers are fixed in size.</td> </tr> <tr> <td>O_PASSOK</td> <td>Validate field only if modified by user.</td> </tr> </table> | O_VISIBLE | The field is displayed. | O_ACTIVE | The field is visited during processing. | O_PUBLIC | The field contents are displayed as data is entered. | O_EDIT | The field can be edited. | O_WRAP | Words not fitting on a line are wrapped to the next line. | O_BLANK | The whole field is cleared if a character is entered in the first position. | O_AUTOSKIP | Skip to the next field when the current field becomes full. | O_NULLOK | A blank field is considered valid. | O_STATIC | The field buffers are fixed in size. | O_PASSOK | Validate field only if modified by user. |
| O_VISIBLE | The field is displayed. | | | | | | | | | | | | | | | | | | | | |
| O_ACTIVE | The field is visited during processing. | | | | | | | | | | | | | | | | | | | | |
| O_PUBLIC | The field contents are displayed as data is entered. | | | | | | | | | | | | | | | | | | | | |
| O_EDIT | The field can be edited. | | | | | | | | | | | | | | | | | | | | |
| O_WRAP | Words not fitting on a line are wrapped to the next line. | | | | | | | | | | | | | | | | | | | | |
| O_BLANK | The whole field is cleared if a character is entered in the first position. | | | | | | | | | | | | | | | | | | | | |
| O_AUTOSKIP | Skip to the next field when the current field becomes full. | | | | | | | | | | | | | | | | | | | | |
| O_NULLOK | A blank field is considered valid. | | | | | | | | | | | | | | | | | | | | |
| O_STATIC | The field buffers are fixed in size. | | | | | | | | | | | | | | | | | | | | |
| O_PASSOK | Validate field only if modified by user. | | | | | | | | | | | | | | | | | | | | |
| RETURN VALUES | <p>set_field_opts, field_opts_on and field_opts_off return one of the following:</p> <table border="0"> <tr> <td style="padding-right: 20px;">E_OK</td> <td>The function returned successfully.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> <tr> <td>E_CURRENT</td> <td>The field is the current field.</td> </tr> </table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. | E_CURRENT | The field is the current field. | | | | | | | | | | | | | | |
| E_OK | The function returned successfully. | | | | | | | | | | | | | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | | | | | | | | | | | | | |
| E_CURRENT | The field is the current field. | | | | | | | | | | | | | | | | | | | | |

field_opts(3CURSES)

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), forms(3CURSES), attributes(5)

NOTES The header `<form.h>` automatically includes the headers `<eti.h>` and `< curses.h>`.

field_opts_off(3CURSES)

| | | | | | | | | | | | | | | | | | | | | | |
|----------------------|---|-----------|-------------------------------------|----------------|---|-----------|--|--------|--------------------------|--------|---|---------|---|------------|---|----------|------------------------------------|----------|--------------------------------------|----------|--|
| NAME | form_field_opts, set_field_opts, field_opts_on, field_opts_off, field_opts – forms field option routines | | | | | | | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_field_opts(FIELD *field, OPTIONS opts); int set_field_opts(FIELD *field, OPTIONS opts); int field_opts_on(FIELD *field, OPTIONS opts); int field_opts_off(FIELD *field, OPTIONS opts); OPTIONS field_opts(FIELD *field);</pre> | | | | | | | | | | | | | | | | | | | | |
| DESCRIPTION | <p>set_field_opts() turns on the named options of <i>field</i> and turns off all remaining options. Options are boolean values that can be OR-ed together.</p> <p>field_opts_on() turns on the named options; no other options are changed.</p> <p>field_opts_off() turns off the named options; no other options are changed.</p> <p>field_opts() returns the options set for <i>field</i>.</p> <table><tr><td>O_VISIBLE</td><td>The field is displayed.</td></tr><tr><td>O_ACTIVE</td><td>The field is visited during processing.</td></tr><tr><td>O_PUBLIC</td><td>The field contents are displayed as data is entered.</td></tr><tr><td>O_EDIT</td><td>The field can be edited.</td></tr><tr><td>O_WRAP</td><td>Words not fitting on a line are wrapped to the next line.</td></tr><tr><td>O_BLANK</td><td>The whole field is cleared if a character is entered in the first position.</td></tr><tr><td>O_AUTOSKIP</td><td>Skip to the next field when the current field becomes full.</td></tr><tr><td>O_NULLOK</td><td>A blank field is considered valid.</td></tr><tr><td>O_STATIC</td><td>The field buffers are fixed in size.</td></tr><tr><td>O_PASSOK</td><td>Validate field only if modified by user.</td></tr></table> | O_VISIBLE | The field is displayed. | O_ACTIVE | The field is visited during processing. | O_PUBLIC | The field contents are displayed as data is entered. | O_EDIT | The field can be edited. | O_WRAP | Words not fitting on a line are wrapped to the next line. | O_BLANK | The whole field is cleared if a character is entered in the first position. | O_AUTOSKIP | Skip to the next field when the current field becomes full. | O_NULLOK | A blank field is considered valid. | O_STATIC | The field buffers are fixed in size. | O_PASSOK | Validate field only if modified by user. |
| O_VISIBLE | The field is displayed. | | | | | | | | | | | | | | | | | | | | |
| O_ACTIVE | The field is visited during processing. | | | | | | | | | | | | | | | | | | | | |
| O_PUBLIC | The field contents are displayed as data is entered. | | | | | | | | | | | | | | | | | | | | |
| O_EDIT | The field can be edited. | | | | | | | | | | | | | | | | | | | | |
| O_WRAP | Words not fitting on a line are wrapped to the next line. | | | | | | | | | | | | | | | | | | | | |
| O_BLANK | The whole field is cleared if a character is entered in the first position. | | | | | | | | | | | | | | | | | | | | |
| O_AUTOSKIP | Skip to the next field when the current field becomes full. | | | | | | | | | | | | | | | | | | | | |
| O_NULLOK | A blank field is considered valid. | | | | | | | | | | | | | | | | | | | | |
| O_STATIC | The field buffers are fixed in size. | | | | | | | | | | | | | | | | | | | | |
| O_PASSOK | Validate field only if modified by user. | | | | | | | | | | | | | | | | | | | | |
| RETURN VALUES | <p>set_field_opts, field_opts_on and field_opts_off return one of the following:</p> <table><tr><td>E_OK</td><td>The function returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr><tr><td>E_CURRENT</td><td>The field is the current field.</td></tr></table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. | E_CURRENT | The field is the current field. | | | | | | | | | | | | | | |
| E_OK | The function returned successfully. | | | | | | | | | | | | | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | | | | | | | | | | | | | |
| E_CURRENT | The field is the current field. | | | | | | | | | | | | | | | | | | | | |

field_opts_off(3CURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `courses(3CURSES)`, `forms(3CURSES)`, `attributes(5)`

NOTES The header `<form.h>` automatically includes the headers `<eti.h>` and `<courses.h>`.

field_opts_on(3CURSES)

| | | | | | | | | | | | | | | | | | | | | | |
|----------------------|---|-----------|-------------------------------------|----------------|---|-----------|--|--------|--------------------------|--------|---|---------|---|------------|---|----------|------------------------------------|----------|--------------------------------------|----------|--|
| NAME | form_field_opts, set_field_opts, field_opts_on, field_opts_off, field_opts – forms field option routines | | | | | | | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_field_opts(FIELD *field, OPTIONS opts); int set_field_opts(FIELD *field, OPTIONS opts); int field_opts_on(FIELD *field, OPTIONS opts); int field_opts_off(FIELD *field, OPTIONS opts); OPTIONS field_opts(FIELD *field);</pre> | | | | | | | | | | | | | | | | | | | | |
| DESCRIPTION | <p>set_field_opts() turns on the named options of <i>field</i> and turns off all remaining options. Options are boolean values that can be OR-ed together.</p> <p>field_opts_on() turns on the named options; no other options are changed.</p> <p>field_opts_off() turns off the named options; no other options are changed.</p> <p>field_opts() returns the options set for <i>field</i>.</p> <table><tr><td>O_VISIBLE</td><td>The field is displayed.</td></tr><tr><td>O_ACTIVE</td><td>The field is visited during processing.</td></tr><tr><td>O_PUBLIC</td><td>The field contents are displayed as data is entered.</td></tr><tr><td>O_EDIT</td><td>The field can be edited.</td></tr><tr><td>O_WRAP</td><td>Words not fitting on a line are wrapped to the next line.</td></tr><tr><td>O_BLANK</td><td>The whole field is cleared if a character is entered in the first position.</td></tr><tr><td>O_AUTOSKIP</td><td>Skip to the next field when the current field becomes full.</td></tr><tr><td>O_NULLOK</td><td>A blank field is considered valid.</td></tr><tr><td>O_STATIC</td><td>The field buffers are fixed in size.</td></tr><tr><td>O_PASSOK</td><td>Validate field only if modified by user.</td></tr></table> | O_VISIBLE | The field is displayed. | O_ACTIVE | The field is visited during processing. | O_PUBLIC | The field contents are displayed as data is entered. | O_EDIT | The field can be edited. | O_WRAP | Words not fitting on a line are wrapped to the next line. | O_BLANK | The whole field is cleared if a character is entered in the first position. | O_AUTOSKIP | Skip to the next field when the current field becomes full. | O_NULLOK | A blank field is considered valid. | O_STATIC | The field buffers are fixed in size. | O_PASSOK | Validate field only if modified by user. |
| O_VISIBLE | The field is displayed. | | | | | | | | | | | | | | | | | | | | |
| O_ACTIVE | The field is visited during processing. | | | | | | | | | | | | | | | | | | | | |
| O_PUBLIC | The field contents are displayed as data is entered. | | | | | | | | | | | | | | | | | | | | |
| O_EDIT | The field can be edited. | | | | | | | | | | | | | | | | | | | | |
| O_WRAP | Words not fitting on a line are wrapped to the next line. | | | | | | | | | | | | | | | | | | | | |
| O_BLANK | The whole field is cleared if a character is entered in the first position. | | | | | | | | | | | | | | | | | | | | |
| O_AUTOSKIP | Skip to the next field when the current field becomes full. | | | | | | | | | | | | | | | | | | | | |
| O_NULLOK | A blank field is considered valid. | | | | | | | | | | | | | | | | | | | | |
| O_STATIC | The field buffers are fixed in size. | | | | | | | | | | | | | | | | | | | | |
| O_PASSOK | Validate field only if modified by user. | | | | | | | | | | | | | | | | | | | | |
| RETURN VALUES | <p>set_field_opts, field_opts_on and field_opts_off return one of the following:</p> <table><tr><td>E_OK</td><td>The function returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr><tr><td>E_CURRENT</td><td>The field is the current field.</td></tr></table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. | E_CURRENT | The field is the current field. | | | | | | | | | | | | | | |
| E_OK | The function returned successfully. | | | | | | | | | | | | | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | | | | | | | | | | | | | |
| E_CURRENT | The field is the current field. | | | | | | | | | | | | | | | | | | | | |

field_opts_on(3CURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `courses(3CURSES)`, `forms(3CURSES)`, `attributes(5)`

NOTES The header `<form.h>` automatically includes the headers `<eti.h>` and `< curses.h>`.

| NAME | form_field_attributes, set_field_fore, field_fore, set_field_back, field_back, set_field_pad, field_pad – format the general display attributes of forms | | | | | | |
|----------------------|--|----------------|-------------------------------------|----------------|---------------|----------------|---------------------------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_field_fore(FIELD *field, chtype attr); chtype field_fore(FIELD *field); int set_field_back(FIELD *field, chtype attr); chtype field_back(FIELD *field); int set_field_pad(FIELD *field, int pad); int field_pad(FIELD *field);</pre> | | | | | | |
| DESCRIPTION | <p>set_field_fore() sets the foreground attribute of <i>field</i>. The foreground attribute is the low-level curses display attribute used to display the field contents. field_fore() returns the foreground attribute of <i>field</i>.</p> <p>set_field_back() sets the background attribute of <i>field</i>. The background attribute is the low-level curses display attribute used to display the extent of the field. field_back() returns the background attribute of <i>field</i>.</p> <p>set_field_pad() sets the pad character of <i>field</i> to <i>pad</i>. The pad character is the character used to fill within the field. field_pad() returns the pad character of <i>field</i>.</p> | | | | | | |
| RETURN VALUES | <p>field_fore(), field_back(), and field_pad() return default values if <i>field</i> is NULL. If <i>field</i> is not NULL and is not a valid FIELD pointer, the return value from these routines is undefined.</p> <p>set_field_fore(), set_field_back(), and set_field_pad() return one of the following:</p> <table border="0"> <tr> <td style="padding-right: 20px;">E_OK</td> <td>The function returned successfully.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> <tr> <td>E_BAD_ARGUMENT</td> <td>An argument is incorrect.</td> </tr> </table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An argument is incorrect. |
| E_OK | The function returned successfully. | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | |
| E_BAD_ARGUMENT | An argument is incorrect. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | | | |

| | | | | | |
|----------------------|---|------|-------------------------------------|----------------|---------------|
| NAME | form_hook, set_form_init, form_init, set_form_term, form_term, set_field_init, field_init, set_field_term, field_term – assign application-specific routines for invocation by forms | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_form_init(FORM *form, void (*func) (FORM*)); void (*form_init) (FORM *form); int set_form_term(FORM *form, void (*func) (FORM*)); void (*form_term) (FORM *form); int set_field_init(FORM *form, void (*func) (FORM*)); void (*field_init) (FORM *form); int set_field_term(FORM *form, void (*func) (FORM*)); void (*field_term) (FORM *form);</pre> | | | | |
| DESCRIPTION | <p>These routines allow the programmer to assign application specific routines to be executed automatically at initialization and termination points in the forms application. The user need not specify any application-defined initialization or termination routines at all, but they may be helpful for displaying messages or page numbers and other chores.</p> <p>set_form_init() assigns an application-defined initialization function to be called when the <i>form</i> is posted and just after a page change. form_init() returns a pointer to the initialization function, if any.</p> <p>set_form_term() assigns an application-defined function to be called when the <i>form</i> is unposted and just before a page change. form_term() returns a pointer to the function, if any.</p> <p>set_field_init() assigns an application-defined function to be called when the <i>form</i> is posted and just after the current field changes. field_init() returns a pointer to the function, if any.</p> <p>set_field_term() assigns an application-defined function to be called when the <i>form</i> is unposted and just before the current field changes. field_term() returns a pointer to the function, if any.</p> | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <table border="0" style="width: 100%;"> <tr> <td style="padding-right: 40px;">E_OK</td> <td>The function returned successfully.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> </table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. |
| E_OK | The function returned successfully. | | | | |
| E_SYSTEM_ERROR | System error. | | | | |

field_term(3CURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `courses(3CURSES)`, `forms(3CURSES)`, `attributes(5)`

NOTES The header `<form.h>` automatically includes the headers `<eti.h>` and `<courses.h>`.

| NAME | form_field_validation, set_field_type, field_type, field_arg – forms field data type validation | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_field_type(FIELD *field, FIELDTYPE *type, ...); FIELDTYPE *field_type(FIELD *field); char *field_arg(FIELD *field);</pre> | | | | |
| DESCRIPTION | <p>set_field_type() associates the specified field type with <i>field</i>. Certain field types take additional arguments. TYPE_ALNUM, for instance, requires one, the minimum width specification for the field. The other predefined field types are: TYPE_ALPHA, TYPE_ENUM, TYPE_INTEGER, TYPE_NUMERIC, and TYPE_REGEX.</p> <p>field_type() returns a pointer to the field type of <i>field</i>. NULL is returned if no field type is assigned.</p> <p>field_arg() returns a pointer to the field arguments associated with the field type of <i>field</i>. NULL is returned if no field type is assigned.</p> | | | | |
| RETURN VALUES | <p>field_type() and field_arg() return NULL on error.</p> <p>set_field_type() returns one of the following:</p> <p>E_OK The function returned successfully.</p> <p>E_SYSTEM_ERROR System error.</p> | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | |

field_userptr(3CURSES)

NAME form_field_userptr, set_field_userptr, field_userptr – associate application data with forms

SYNOPSIS

```
cc [ flag... ] file... -lform -lcurses [ library... ]
#include <form.h>

int set_field_userptr(FIELD *field, char *ptr);
char *field_userptr(FIELD *field);
```

DESCRIPTION Every field has an associated user pointer that can be used to store pertinent data. set_field_userptr() sets the user pointer of *field*. field_userptr() returns the user pointer of *field*.

RETURN VALUES field_userptr() returns NULL on error. set_field_userptr() returns one of the following:

| | |
|----------------|-------------------------------------|
| E_OK | The function returned successfully. |
| E_SYSTEM_ERROR | System error. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), forms(3CURSES), attributes(5)

NOTES The header <form.h> automatically includes the headers <eti.h> and <curses.h>.

| | |
|--------------------|--|
| NAME | curs_util, unctrl, keyname, filter, use_env, putwin, getwin, delay_output, flushinp – curses miscellaneous utility routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> char *unctrl (chtype <i>c</i>); char *keyname (int <i>c</i>); int filter (void); void use_env (char <i>bool</i>); int putwin (WINDOW *<i>win</i>, FILE *<i>filep</i>); WINDOW *getwin (FILE *<i>filep</i>); int delay_output (int <i>ms</i>); int flushinp (void);</pre> |
| DESCRIPTION | <p>The <code>unctrl()</code> macro expands to a character string which is a printable representation of the character <code>c</code>. Control characters are displayed in the <code>^X</code> notation. Printing characters are displayed as is.</p> <p>With the <code>keyname()</code> routine, a character string corresponding to the key <code>c</code> is returned.</p> <p>The <code>filter()</code> routine, if used, is called before <code>initscr()</code> or <code>newterm()</code> are called. It makes <code>curses</code> think that there is a one-line screen. <code>curses</code> does not use any terminal capabilities that assume that they know on what line of the screen the cursor is positioned.</p> <p>The <code>use_env()</code> routine, if used, is called before <code>initscr()</code> or <code>newterm()</code> are called. When called with <code>FALSE</code> as an argument, the values of <code>lines</code> and <code>columns</code> specified in the <code>terminfo</code> database will be used, even if environment variables <code>LINES</code> and <code>COLUMNS</code> (used by default) are set, or if <code>curses</code> is running in a window (in which case default behavior would be to use the window size if <code>LINES</code> and <code>COLUMNS</code> are not set).</p> <p>With the <code>putwin()</code> routine, all data associated with window <code>win</code> is written into the file to which <code>filep</code> points. This information can be later retrieved using the <code>getwin()</code> function.</p> <p>The <code>getwin()</code> routine reads window related data stored in the file by <code>putwin()</code>. The routine then creates and initializes a new window using that data. It returns a pointer to the new window.</p> <p>The <code>delay_output()</code> routine inserts an <code>ms</code> millisecond pause in output. This routine should not be used extensively because padding characters are used rather than a CPU pause.</p> <p>The <code>flushinp()</code> routine throws away any typeahead that has been typed by the user and has not yet been read by the program.</p> |

filter(3CURSES)

RETURN VALUES Except for `flushinp()`, routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`flushinp()` always returns `OK`.

Routines that return pointers return `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_initscr(3CURSES)`, `curs_scr_dump(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `unctrl()` is a macro, which is defined in `<unctrl.h>`.

| NAME | filter – disable use of certain terminal capabilities | | | | | | |
|----------------------|--|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> void filter(void);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>filter()</code> function changes how X/Open Curses initializes terminal capabilities that assume the terminal has more than one line. After a call to <code>filter()</code>, the <code>initscr(3XCURSES)</code> or <code>newterm(3XCURSES)</code> functions also:</p> <ul style="list-style-type: none"> ■ Disable use of <code>clear</code>, <code>cu</code>, <code>cu1</code>, <code>cuu1</code>, and <code>vpa</code>. ■ Set home string to the value of <code>cr</code>. ■ Set lines to 1. | | | | | | |
| RETURN VALUES | The <code>filter()</code> function does not return a value. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>initscr(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>newterm(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

flash(3CURSES)

| NAME | <code> curs_beep</code> , <code> beep</code> , <code> flash</code> – curses bell and screen flash routines | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int beep(void); int flash(void);</pre> | | | | |
| DESCRIPTION | The <code> beep()</code> and <code> flash()</code> routines are used to signal the terminal user. The routine <code> beep()</code> sounds the audible alarm on the terminal, if possible; if that is not possible, it flashes the screen (visible bell), if that is possible. The routine <code> flash()</code> flashes the screen, and if that is not possible, sounds the audible signal. If neither signal is possible, nothing happens. Nearly all terminals have an audible signal (bell or beep), but only some can flash the screen. | | | | |
| RETURN VALUES | These routines always return OK. | | | | |
| ATTRIBUTES | See <code> attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code> curses(3CURSES)</code> , <code> attributes(5)</code> | | | | |
| NOTES | The header <code> <curses.h></code> automatically includes the headers <code> <stdio.h></code> and <code> <unctrl.h></code> . | | | | |

| NAME | beep, flash – activate audio-visual alarm | | | | | | |
|----------------------|--|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int beep(void); int flash(void);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>beep()</code> and <code>flash()</code> functions produce an audio and visual alarm on the terminal, respectively. If the terminal has the capability, <code>beep()</code> sounds a bell or beep and <code>flash()</code> flashes the screen. One alarm is substituted for another if the terminal does not support the capability called (see <code>terminfo(4)</code> <code>bel</code> and <code>flash</code> capabilities). For example, a call to <code>beep()</code> for a terminal without that capability results in a flash.</p> | | | | | | |
| RETURN VALUES | These functions always return OK. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>libcurses(3XCURSES)</code> , <code>terminfo(4)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

flushinp(3CURSES)

| | |
|--------------------|---|
| NAME | <code>curs_util</code> , <code>unctrl</code> , <code>keyname</code> , <code>filter</code> , <code>use_env</code> , <code>putwin</code> , <code>getwin</code> , <code>delay_output</code> , <code>flushinp</code> – curses miscellaneous utility routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> char *unctrl (chtype <i>c</i>); char *keyname (int <i>c</i>); int filter (void); void use_env (char <i>bool</i>); int putwin (WINDOW *<i>win</i>, FILE *<i>filep</i>); WINDOW *getwin (FILE *<i>filep</i>); int delay_output (int <i>ms</i>); int flushinp (void);</pre> |
| DESCRIPTION | <p>The <code>unctrl()</code> macro expands to a character string which is a printable representation of the character <i>c</i>. Control characters are displayed in the ^X notation. Printing characters are displayed as is.</p> <p>With the <code>keyname()</code> routine, a character string corresponding to the key <i>c</i> is returned.</p> <p>The <code>filter()</code> routine, if used, is called before <code>initscr()</code> or <code>newterm()</code> are called. It makes <code>curses</code> think that there is a one-line screen. <code>curses</code> does not use any terminal capabilities that assume that they know on what line of the screen the cursor is positioned.</p> <p>The <code>use_env()</code> routine, if used, is called before <code>initscr()</code> or <code>newterm()</code> are called. When called with <code>FALSE</code> as an argument, the values of <code>lines</code> and <code>columns</code> specified in the <code>terminfo</code> database will be used, even if environment variables <code>LINES</code> and <code>COLUMNS</code> (used by default) are set, or if <code>curses</code> is running in a window (in which case default behavior would be to use the window size if <code>LINES</code> and <code>COLUMNS</code> are not set).</p> <p>With the <code>putwin()</code> routine, all data associated with window <i>win</i> is written into the file to which <i>filep</i> points. This information can be later retrieved using the <code>getwin()</code> function.</p> <p>The <code>getwin()</code> routine reads window related data stored in the file by <code>putwin()</code>. The routine then creates and initializes a new window using that data. It returns a pointer to the new window.</p> <p>The <code>delay_output()</code> routine inserts an <i>ms</i> millisecond pause in output. This routine should not be used extensively because padding characters are used rather than a CPU pause.</p> <p>The <code>flushinp()</code> routine throws away any typeahead that has been typed by the user and has not yet been read by the program.</p> |

flushinp(3CURSES)

RETURN VALUES Except for `flushinp()`, routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`flushinp()` always returns `OK`.

Routines that return pointers return `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_initscr(3CURSES)`, `curl_scr_dump(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `unctrl()` is a macro, which is defined in `<unctrl.h>`.

flushinp(3XCURSES)

NAME flushinp – discard type-ahead characters

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

int flushinp(void);
```

DESCRIPTION The flushinp() function discards (flushes) any characters in the input buffer associated with the current screen.

RETURN VALUES The flushinp() function always returns OK.

ERRORS No errors are defined.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO libcurses(3XCURSES), attributes(5), standards(5)

NAME form_cursor, pos_form_cursor – position forms window cursor

SYNOPSIS

```
cc [ flag... ] file... -lform -lcurses [ library... ]
#include <form.h>

int pos_form_cursor(FORM *form);
```

DESCRIPTION pos_form_cursor() moves the form window cursor to the location required by the form driver to resume form processing. This may be needed after the application calls a curses library I/O routine.

RETURN VALUES pos_form_cursor() returns one of the following:

| | |
|----------------|-------------------------------------|
| E_OK | The function returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An argument is incorrect. |
| E_NOT_POSTED | The form is not posted. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), forms(3CURSES), attributes(5)

NOTES The header <form.h> automatically includes the headers <eti.h> and <curses.h>.

form_data(3CURSES)

| NAME | form_data, data_ahead, data_behind – tell if forms field has off-screen data ahead or behind | | | | |
|--------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lform -lcurses [library ..] #include <form.h> int data_ahead(FORM *form) ; int data_behind(FORM *form) ;</pre> | | | | |
| DESCRIPTION | <p>data_ahead() returns TRUE (1) if the current field has more off-screen data ahead; otherwise it returns FALSE (0).</p> <p>data_behind() returns TRUE (1) if the current field has more off-screen data behind; otherwise it returns FALSE (0).</p> | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--------------------|---|---------------|------------------------|---------------|----------------------------|----------------|-------------------------|---------------|------------------------|----------------|-------------------------|----------------|-----------------------------|-----------------|--------------------------|----------------|-------------------------|-----------------|--------------------------------|-----------------|--------------------------------|------------------|---------------------------------|-----------------|--------------------------------|----------------|---------------------|-----------------|----------------------|--------------|-------------------|----------------|---------------------|---------------|--|---------------|--|---------------|-------------------------------------|---------------|---|---------------|-------------------------------------|
| NAME | form_driver – command processor for the forms subsystem | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int form_driver(FORM *form, int c);</pre> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DESCRIPTION | <p>The <code>form_driver()</code> function is the workhorse of the forms subsystem; it checks to determine whether the character <code>c</code> is a forms request or data. If it is a request, the form driver executes the request and reports the result. If it is data (a printable ASCII character), it enters the data into the current position in the current field. If it is not recognized, the form driver assumes it is an application-defined command and returns <code>E_UNKNOWN_COMMAND</code>. Application defined commands should be defined relative to <code>MAX_COMMAND</code>, the maximum value of a request listed below.</p> <p>Form driver requests:</p> <table border="0"> <tr><td>REQ_NEXT_PAGE</td><td>Move to the next page.</td></tr> <tr><td>REQ_PREV_PAGE</td><td>Move to the previous page.</td></tr> <tr><td>REQ_FIRST_PAGE</td><td>Move to the first page.</td></tr> <tr><td>REQ_LAST_PAGE</td><td>Move to the last page.</td></tr> <tr><td>REQ_NEXT_FIELD</td><td>Move to the next field.</td></tr> <tr><td>REQ_PREV_FIELD</td><td>Move to the previous field.</td></tr> <tr><td>REQ_FIRST_FIELD</td><td>Move to the first field.</td></tr> <tr><td>REQ_LAST_FIELD</td><td>Move to the last field.</td></tr> <tr><td>REQ_SNEXT_FIELD</td><td>Move to the sorted next field.</td></tr> <tr><td>REQ_SPREV_FIELD</td><td>Move to the sorted prev field.</td></tr> <tr><td>REQ_SFIRST_FIELD</td><td>Move to the sorted first field.</td></tr> <tr><td>REQ_SLAST_FIELD</td><td>Move to the sorted last field.</td></tr> <tr><td>REQ_LEFT_FIELD</td><td>Move left to field.</td></tr> <tr><td>REQ_RIGHT_FIELD</td><td>Move right to field.</td></tr> <tr><td>REQ_UP_FIELD</td><td>Move up to field.</td></tr> <tr><td>REQ_DOWN_FIELD</td><td>Move down to field.</td></tr> <tr><td>REQ_NEXT_CHAR</td><td>Move to the next character in the field.</td></tr> <tr><td>REQ_PREV_CHAR</td><td>Move to the previous character in the field.</td></tr> <tr><td>REQ_NEXT_LINE</td><td>Move to the next line in the field.</td></tr> <tr><td>REQ_PREV_LINE</td><td>Move to the previous line in the field.</td></tr> <tr><td>REQ_NEXT_WORD</td><td>Move to the next word in the field.</td></tr> </table> | REQ_NEXT_PAGE | Move to the next page. | REQ_PREV_PAGE | Move to the previous page. | REQ_FIRST_PAGE | Move to the first page. | REQ_LAST_PAGE | Move to the last page. | REQ_NEXT_FIELD | Move to the next field. | REQ_PREV_FIELD | Move to the previous field. | REQ_FIRST_FIELD | Move to the first field. | REQ_LAST_FIELD | Move to the last field. | REQ_SNEXT_FIELD | Move to the sorted next field. | REQ_SPREV_FIELD | Move to the sorted prev field. | REQ_SFIRST_FIELD | Move to the sorted first field. | REQ_SLAST_FIELD | Move to the sorted last field. | REQ_LEFT_FIELD | Move left to field. | REQ_RIGHT_FIELD | Move right to field. | REQ_UP_FIELD | Move up to field. | REQ_DOWN_FIELD | Move down to field. | REQ_NEXT_CHAR | Move to the next character in the field. | REQ_PREV_CHAR | Move to the previous character in the field. | REQ_NEXT_LINE | Move to the next line in the field. | REQ_PREV_LINE | Move to the previous line in the field. | REQ_NEXT_WORD | Move to the next word in the field. |
| REQ_NEXT_PAGE | Move to the next page. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_PREV_PAGE | Move to the previous page. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_FIRST_PAGE | Move to the first page. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_LAST_PAGE | Move to the last page. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_NEXT_FIELD | Move to the next field. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_PREV_FIELD | Move to the previous field. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_FIRST_FIELD | Move to the first field. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_LAST_FIELD | Move to the last field. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_SNEXT_FIELD | Move to the sorted next field. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_SPREV_FIELD | Move to the sorted prev field. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_SFIRST_FIELD | Move to the sorted first field. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_SLAST_FIELD | Move to the sorted last field. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_LEFT_FIELD | Move left to field. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_RIGHT_FIELD | Move right to field. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_UP_FIELD | Move up to field. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_DOWN_FIELD | Move down to field. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_NEXT_CHAR | Move to the next character in the field. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_PREV_CHAR | Move to the previous character in the field. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_NEXT_LINE | Move to the next line in the field. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_PREV_LINE | Move to the previous line in the field. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_NEXT_WORD | Move to the next word in the field. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

form_driver(3CURSES)

| | |
|----------------|---|
| REQ_PREV_WORD | Move to the previous word in the field. |
| REQ_BEG_FIELD | Move to the first char in the field. |
| REQ_END_FIELD | Move after the last char in the field. |
| REQ_BEG_LINE | Move to the beginning of the line. |
| REQ_END_LINE | Move after the last char in the line. |
| REQ_LEFT_CHAR | Move left in the field. |
| REQ_RIGHT_CHAR | Move right in the field. |
| REQ_UP_CHAR | Move up in the field. |
| REQ_DOWN_CHAR | Move down in the field. |
| REQ_NEW_LINE | Insert/overlay a new line. |
| REQ_INS_CHAR | Insert the blank character at the cursor. |
| REQ_INS_LINE | Insert a blank line at the cursor. |
| REQ_DEL_CHAR | Delete the character at the cursor. |
| REQ_DEL_PREV | Delete the character before the cursor. |
| REQ_DEL_LINE | Delete the line at the cursor. |
| REQ_DEL_WORD | Delete the word at the cursor. |
| REQ_CLR_EOL | Clear to the end of the line. |
| REQ_CLR_EOF | Clear to the end of the field. |
| REQ_CLR_FIELD | Clear the entire field. |
| REQ_OVL_MODE | Enter overlay mode. |
| REQ_INS_MODE | Enter insert mode. |
| REQ_SCR_FLINE | Scroll the field forward a line. |
| REQ_SCR_BLINE | Scroll the field backward a line. |
| REQ_SCR_FPAGE | Scroll the field forward a page. |
| REQ_SCR_BPAGE | Scroll the field backward a page. |
| REQ_SCR_FHPAGE | Scroll the field forward half a page. |
| REQ_SCR_BHPAGE | Scroll the field backward half a page. |
| REQ_SCR_FCHAR | Horizontal scroll forward a character. |
| REQ_SCR_BCHAR | Horizontal scroll backward a character. |
| REQ_SCR_HFLINE | Horizontal scroll forward a line. |
| REQ_SCR_HBLINE | Horizontal scroll backward a line. |

form_driver(3CURSES)

REQ_SCR_HFHALL Horizontal scroll forward half a line.
REQ_SCR_HBHALF Horizontal scroll backward half a line.
REQ_VALIDATION Validate field.
REQ_PREV_CHOICE Display the previous field choice.
REQ_NEXT_CHOICE Display the next field choice.

RETURN VALUES

The `form_driver()` function returns one of the following:

E_OK The function returned successfully.
E_SYSTEM_ERROR System error.
E_BAD_ARGUMENT An argument is incorrect.
E_NOT_POSTED The form is not posted.
E_INVALID_FIELD The field contents are invalid.
E_BAD_STATE The routine was called from an initialization or termination function.
E_REQUEST_DENIED The form driver request failed.
E_UNKNOWN_COMMAND An unknown request was passed to the form driver.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curses(3CURSES)`, `forms(3CURSES)`, `attributes(5)`

NOTES

The header `<form.h>` automatically includes the headers `<eti.h>` and `<curses.h>`.

form_field(3CURSES)

NAME | form_field, set_form_fields, form_fields, field_count, move_field – connect fields to forms

SYNOPSIS | `cc [flag...] file... -lform -lcurses [library...]
#include <form.h>`

| `int set_form_fields(FORM *form, FIELD **field);`
| `FIELD **form_fields(FORM *form);`
| `int field_count(FORM *form);`
| `int move_field(FIELD *field, int frow, int fcol);`

DESCRIPTION | `set_form_fields()` changes the fields connected to *form* to *fields*. The original fields are disconnected.

| `form_fields()` returns a pointer to the field pointer array connected to *form*.

| `field_count()` returns the number of fields connected to *form*.

| `move_field()` moves the disconnected *field* to the location *frow*, *fcol* in the forms subwindow.

RETURN VALUES | `form_fields()` returns NULL on error.

| `field_count()` returns -1 on error.

| `set_form_fields()` and `move_field()` return one of the following:

E_OK The function returned successfully.

E_CONNECTED The field is already connected to a form.

E_SYSTEM_ERROR System error.

E_BAD_ARGUMENT An argument is incorrect

E_POSTED The form is posted.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | `curses(3CURSES)`, `forms(3CURSES)`, `attributes(5)`

NOTES | The header `<form.h>` automatically includes the headers `<eti.h>` and `<curses.h>`.

NAME form_field_attributes, set_field_fore, field_fore, set_field_back, field_back, set_field_pad, field_pad – format the general display attributes of forms

SYNOPSIS

```
cc [ flag... ] file... -lform -lcurses [ library... ]
#include <form.h>

int set_field_fore(FIELD *field, chtype attr);
chtype field_fore(FIELD *field);

int set_field_back(FIELD *field, chtype attr);
chtype field_back(FIELD *field);

int set_field_pad(FIELD *field, int pad);
int field_pad(FIELD *field);
```

DESCRIPTION

set_field_fore() sets the foreground attribute of *field*. The foreground attribute is the low-level curses display attribute used to display the field contents. field_fore() returns the foreground attribute of *field*.

set_field_back() sets the background attribute of *field*. The background attribute is the low-level curses display attribute used to display the extent of the field. field_back() returns the background attribute of *field*.

set_field_pad() sets the pad character of *field* to *pad*. The pad character is the character used to fill within the field. field_pad() returns the pad character of *field*.

RETURN VALUES

field_fore(), field_back(), and field_pad() return default values if *field* is NULL. If *field* is not NULL and is not a valid FIELD pointer, the return value from these routines is undefined.

set_field_fore(), set_field_back(), and set_field_pad() return one of the following:

| | |
|----------------|-------------------------------------|
| E_OK | The function returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An argument is incorrect. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), forms(3CURSES), attributes(5)

NOTES The header <form.h> automatically includes the headers <eti.h> and <curses.h>.

form_field_buffer(3CURSES)

| NAME | form_field_buffer, set_field_buffer, field_buffer, set_field_status, field_status, set_max_field – set and get forms field attributes | | | | | | |
|----------------------|---|----------------|-------------------------------------|----------------|--------------|----------------|---------------------------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_field_buffer(FIELD *field, int buf, char *value); char *field_buffer(FIELD *field, int buf); int set_field_status(FIELD *field, int status); int field_status(FIELD *field); int set_max_field(FIELD *field, int max);</pre> | | | | | | |
| DESCRIPTION | <p>set_field_buffer() sets buffer <i>buf</i> of <i>field</i> to <i>value</i>. Buffer 0 stores the displayed contents of the field. Buffers other than 0 are application specific and not used by the forms library routines. field_buffer() returns the value of <i>field</i> buffer <i>buf</i>.</p> <p>Every field has an associated status flag that is set whenever the contents of field buffer 0 changes. set_field_status() sets the status flag of <i>field</i> to <i>status</i>. field_status() returns the status of <i>field</i>.</p> <p>set_max_field() sets a maximum growth on a dynamic field, or if <i>max</i>=0 turns off any maximum growth.</p> | | | | | | |
| RETURN VALUES | <p>field_buffer() returns NULL on error.</p> <p>field_status() returns TRUE or FALSE.</p> <p>set_field_buffer(), set_field_status(), and set_max_field() return one of the following:</p> <table border="0" style="margin-left: 20px;"> <tr> <td style="padding-right: 20px;">E_OK</td> <td>The function returned successfully.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error</td> </tr> <tr> <td>E_BAD_ARGUMENT</td> <td>An argument is incorrect.</td> </tr> </table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error | E_BAD_ARGUMENT | An argument is incorrect. |
| E_OK | The function returned successfully. | | | | | | |
| E_SYSTEM_ERROR | System error | | | | | | |
| E_BAD_ARGUMENT | An argument is incorrect. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th style="padding: 5px;">ATTRIBUTE TYPE</th> <th style="padding: 5px;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td style="padding: 5px;">MT-Level</td> <td style="padding: 5px;">Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | | | |

NAME form_field_info, field_info, dynamic_field_info – get forms field characteristics

SYNOPSIS

```
cc [ flag... ] file... -lform -lcurses [ library... ]
#include <form.h>

int field_info(FIELD *field, int *rows, int *cols, int *frow, int *fcol,
              int *nrow, int *nbuf);

int dynamic_field_info(FIELD *field, int *drows, int *dcols, int
                      *max);
```

DESCRIPTION

field_info() returns the size, position, and other named field characteristics, as defined in the original call to new_field(), to the locations pointed to by the arguments rows, cols, frow, fcol, nrow, and nbuf.

dynamic_field_info() returns the actual size of the field in the pointer arguments drows, dcols and returns the maximum growth allowed for field in max. If no maximum growth limit is specified for field, max will contain 0. A field can be made dynamic by turning off the field option O_STATIC.

RETURN VALUES These routines return one of the following:

| | |
|----------------|-------------------------------------|
| E_OK | The function returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An argument is incorrect. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), forms(3CURSES), attributes(5)

NOTES The header <form.h> automatically includes the headers <eti.h> and <curses.h>.

form_field_new(3CURSES)

NAME form_field_new, new_field, dup_field, link_field, free_field – create and destroy forms fields

SYNOPSIS

```
cc [ flag... ] file... -lform -lcurses [ library... ]
#include <form.h>

FIELD *new_field(int r, int c, int frow, int fcol, int nrow, int ncol);
FIELD *dup_field(FIELD *field, int frow, int fcol);
FIELD *link_field(FIELD *field, int frow, int fcol);
int free_field(FIELD *field);
```

DESCRIPTION

new_field() creates a new field with *r* rows and *c* columns, starting at *frow*, *fcol*, in the subwindow of a form. *nrow* is the number of off-screen rows and *nbuf* is the number of additional working buffers. This routine returns a pointer to the new field.

dup_field() duplicates *field* at the specified location. All field attributes are duplicated, including the current contents of the field buffers.

link_field() also duplicates *field* at the specified location. However, unlike dup_field(), the new field shares the field buffers with the original field. After creation, the attributes of the new field can be changed without affecting the original field.

free_field() frees the storage allocated for *field*.

RETURN VALUES

Routines that return pointers return NULL on error. free_field() returns one of the following:

| | |
|----------------|---|
| E_OK | The function returned successfully. |
| E_CONNECTED | The field is already connected to a form. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An argument is incorrect. |

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), forms(3CURSES), attributes(5)

NOTES

The header <form.h> automatically includes the headers <eti.h> and <curses.h>.

form_field_opts(3CURSES)

| | | | | | | | | | | | | | | | | | | | | | |
|----------------------|---|-----------|-------------------------------------|----------------|---|-----------|--|--------|--------------------------|--------|---|---------|---|------------|---|----------|------------------------------------|----------|--------------------------------------|----------|--|
| NAME | form_field_opts, set_field_opts, field_opts_on, field_opts_off, field_opts – forms field option routines | | | | | | | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_field_opts(FIELD *field, OPTIONS opts); int set_field_opts(FIELD *field, OPTIONS opts); int field_opts_on(FIELD *field, OPTIONS opts); int field_opts_off(FIELD *field, OPTIONS opts); OPTIONS field_opts(FIELD *field);</pre> | | | | | | | | | | | | | | | | | | | | |
| DESCRIPTION | <p>set_field_opts() turns on the named options of <i>field</i> and turns off all remaining options. Options are boolean values that can be OR-ed together.</p> <p>field_opts_on() turns on the named options; no other options are changed.</p> <p>field_opts_off() turns off the named options; no other options are changed.</p> <p>field_opts() returns the options set for <i>field</i>.</p> <table><tr><td>O_VISIBLE</td><td>The field is displayed.</td></tr><tr><td>O_ACTIVE</td><td>The field is visited during processing.</td></tr><tr><td>O_PUBLIC</td><td>The field contents are displayed as data is entered.</td></tr><tr><td>O_EDIT</td><td>The field can be edited.</td></tr><tr><td>O_WRAP</td><td>Words not fitting on a line are wrapped to the next line.</td></tr><tr><td>O_BLANK</td><td>The whole field is cleared if a character is entered in the first position.</td></tr><tr><td>O_AUTOSKIP</td><td>Skip to the next field when the current field becomes full.</td></tr><tr><td>O_NULLOK</td><td>A blank field is considered valid.</td></tr><tr><td>O_STATIC</td><td>The field buffers are fixed in size.</td></tr><tr><td>O_PASSOK</td><td>Validate field only if modified by user.</td></tr></table> | O_VISIBLE | The field is displayed. | O_ACTIVE | The field is visited during processing. | O_PUBLIC | The field contents are displayed as data is entered. | O_EDIT | The field can be edited. | O_WRAP | Words not fitting on a line are wrapped to the next line. | O_BLANK | The whole field is cleared if a character is entered in the first position. | O_AUTOSKIP | Skip to the next field when the current field becomes full. | O_NULLOK | A blank field is considered valid. | O_STATIC | The field buffers are fixed in size. | O_PASSOK | Validate field only if modified by user. |
| O_VISIBLE | The field is displayed. | | | | | | | | | | | | | | | | | | | | |
| O_ACTIVE | The field is visited during processing. | | | | | | | | | | | | | | | | | | | | |
| O_PUBLIC | The field contents are displayed as data is entered. | | | | | | | | | | | | | | | | | | | | |
| O_EDIT | The field can be edited. | | | | | | | | | | | | | | | | | | | | |
| O_WRAP | Words not fitting on a line are wrapped to the next line. | | | | | | | | | | | | | | | | | | | | |
| O_BLANK | The whole field is cleared if a character is entered in the first position. | | | | | | | | | | | | | | | | | | | | |
| O_AUTOSKIP | Skip to the next field when the current field becomes full. | | | | | | | | | | | | | | | | | | | | |
| O_NULLOK | A blank field is considered valid. | | | | | | | | | | | | | | | | | | | | |
| O_STATIC | The field buffers are fixed in size. | | | | | | | | | | | | | | | | | | | | |
| O_PASSOK | Validate field only if modified by user. | | | | | | | | | | | | | | | | | | | | |
| RETURN VALUES | <p>set_field_opts, field_opts_on and field_opts_off return one of the following:</p> <table><tr><td>E_OK</td><td>The function returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr><tr><td>E_CURRENT</td><td>The field is the current field.</td></tr></table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. | E_CURRENT | The field is the current field. | | | | | | | | | | | | | | |
| E_OK | The function returned successfully. | | | | | | | | | | | | | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | | | | | | | | | | | | | |
| E_CURRENT | The field is the current field. | | | | | | | | | | | | | | | | | | | | |

`form_field_opts(3CURSES)`

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3CURSES)`, `forms(3CURSES)`, `attributes(5)`

NOTES The header `<form.h>` automatically includes the headers `<eti.h>` and `< curses.h>`.

form_fields(3CURSES)

| NAME | form_field, set_form_fields, form_fields, field_count, move_field – connect fields to forms | | | | | | | | | | |
|----------------------|--|----------------|-------------------------------------|-------------|---|----------------|---------------|----------------|--------------------------|----------|---------------------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_form_fields(FORM *form, FIELD **field); FIELD **form_fields(FORM *form); int field_count(FORM *form); int move_field(FIELD *field, int frow, int fcol);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>set_form_fields() changes the fields connected to <i>form</i> to <i>fields</i>. The original fields are disconnected.</p> <p>form_fields() returns a pointer to the field pointer array connected to <i>form</i>.</p> <p>field_count() returns the number of fields connected to <i>form</i>.</p> <p>move_field() moves the disconnected <i>field</i> to the location <i>frow</i>, <i>fcol</i> in the forms subwindow.</p> | | | | | | | | | | |
| RETURN VALUES | <p>form_fields() returns NULL on error.</p> <p>field_count() returns -1 on error.</p> <p>set_form_fields() and move_field() return one of the following:</p> <table><tr><td>E_OK</td><td>The function returned successfully.</td></tr><tr><td>E_CONNECTED</td><td>The field is already connected to a form.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr><tr><td>E_BAD_ARGUMENT</td><td>An argument is incorrect</td></tr><tr><td>E_POSTED</td><td>The form is posted.</td></tr></table> | E_OK | The function returned successfully. | E_CONNECTED | The field is already connected to a form. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An argument is incorrect | E_POSTED | The form is posted. |
| E_OK | The function returned successfully. | | | | | | | | | | |
| E_CONNECTED | The field is already connected to a form. | | | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | | | |
| E_BAD_ARGUMENT | An argument is incorrect | | | | | | | | | | |
| E_POSTED | The form is posted. | | | | | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | | | | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | | | | | |
| MT-Level | Unsafe | | | | | | | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | | | | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | | | | | | | |

| | | | | | | | | | |
|----------------------|--|------|-------------------------------------|----------------|---------------|----------------|---------------------------|-------------|--|
| NAME | form_fieldtype, new_fieldtype, free_fieldtype, set_fieldtype_arg, set_fieldtype_choice, link_fieldtype – forms fieldtype routines | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> FIELDTYPE *new_fieldtype(int (* field_check)(FIELD *, char *), int (*char_check)(int, char *)); int free_fieldtype(FIELDTYPE *fieldtype); int set_fieldtype_arg(FIELDTYPE *fieldtype, char *(* mak_arg)(va_list *), char *(* copy_arg)(char *), void (* free_arg)(char *)); int set_fieldtype_choice(FIELDTYPE *fieldtype, int (* next_choice)(FIELD *, char *), int (*prev_choice)(FIELD *, char *)); FIELDTYPE *link_fieldtype(FIELDTYPE *type1, FIELDTYPE *type2);</pre> | | | | | | | | |
| DESCRIPTION | <p>new_fieldtype() creates a new field type. The application programmer must write the function <i>field_check</i>, which validates the field value, and the function <i>char_check</i>, which validates each character. free_fieldtype() frees the space allocated for the field type.</p> <p>By associating function pointers with a field type, set_fieldtype_arg() connects to the field type additional arguments necessary for a set_field_type() call. Function <i>mak_arg</i> allocates a structure for the field specific parameters to set_field_type() and returns a pointer to the saved data. Function <i>copy_arg</i> duplicates the structure created by <i>make_arg</i>. Function <i>free_arg</i> frees any storage allocated by <i>make_arg</i> or <i>copy_arg</i>.</p> <p>The form_driver() requests REQ_NEXT_CHOICE and REQ_PREV_CHOICE let the user request the next or previous value of a field type comprising an ordered set of values. set_fieldtype_choice() allows the application programmer to implement these requests for the given field type. It associates with the given field type those application-defined functions that return pointers to the next or previous choice for the field.</p> <p>link_fieldtype() returns a pointer to the field type built from the two given types. The constituent types may be any application-defined or pre-defined types.</p> | | | | | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <table border="0"> <tr> <td style="padding-right: 20px;">E_OK</td> <td>The function returned successfully.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> <tr> <td>E_BAD_ARGUMENT</td> <td>An argument is incorrect.</td> </tr> <tr> <td>E_CONNECTED</td> <td>Type is connected to one or more fields.</td> </tr> </table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An argument is incorrect. | E_CONNECTED | Type is connected to one or more fields. |
| E_OK | The function returned successfully. | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | |
| E_BAD_ARGUMENT | An argument is incorrect. | | | | | | | | |
| E_CONNECTED | Type is connected to one or more fields. | | | | | | | | |

form_fieldtype(3CURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `courses(3CURSES)`, `forms(3CURSES)`, `attributes(5)`

NOTES The header `<form.h>` automatically includes the headers `<eti.h>` and `< curses.h>`.

form_field_validation(3CURSES)

| NAME | form_field_validation, set_field_type, field_type, field_arg – forms field data type validation | | | | |
|----------------------|---|----------------|-------------------------------------|----------------|---------------|
| SYNOPSIS | <pre>cc [<i>flag...</i>] <i>file...</i> -lform -lcurses [<i>library...</i>] #include <form.h> int set_field_type(FIELD *<i>field</i>, FIELDTYPE *<i>type</i>, ...); FIELDTYPE *field_type(FIELD *<i>field</i>); char *field_arg(FIELD *<i>field</i>);</pre> | | | | |
| DESCRIPTION | <p>set_field_type() associates the specified field type with <i>field</i>. Certain field types take additional arguments. TYPE_ALNUM, for instance, requires one, the minimum width specification for the field. The other predefined field types are: TYPE_ALPHA, TYPE_ENUM, TYPE_INTEGER, TYPE_NUMERIC, and TYPE_REGEXP.</p> <p>field_type() returns a pointer to the field type of <i>field</i>. NULL is returned if no field type is assigned.</p> <p>field_arg() returns a pointer to the field arguments associated with the field type of <i>field</i>. NULL is returned if no field type is assigned.</p> | | | | |
| RETURN VALUES | <p>field_type() and field_arg() return NULL on error.</p> <p>set_field_type() returns one of the following:</p> <table><tr><td>E_OK</td><td>The function returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr></table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. |
| E_OK | The function returned successfully. | | | | |
| E_SYSTEM_ERROR | System error. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | |

| | | | | | |
|----------------------|---|------|-------------------------------------|----------------|---------------|
| NAME | form_hook, set_form_init, form_init, set_form_term, form_term, set_field_init, field_init, set_field_term, field_term – assign application-specific routines for invocation by forms | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_form_init(FORM *form, void (*func) (FORM*)); void (*form_init) (FORM *form); int set_form_term(FORM *form, void (*func) (FORM*)); void (*form_term) (FORM *form); int set_field_init(FORM *form, void (*func) (FORM*)); void (*field_init) (FORM *form); int set_field_term(FORM *form, void (*func) (FORM*)); void (*field_term) (FORM *form);</pre> | | | | |
| DESCRIPTION | <p>These routines allow the programmer to assign application specific routines to be executed automatically at initialization and termination points in the forms application. The user need not specify any application-defined initialization or termination routines at all, but they may be helpful for displaying messages or page numbers and other chores.</p> <p>set_form_init() assigns an application-defined initialization function to be called when the <i>form</i> is posted and just after a page change. form_init() returns a pointer to the initialization function, if any.</p> <p>set_form_term() assigns an application-defined function to be called when the <i>form</i> is unposted and just before a page change. form_term() returns a pointer to the function, if any.</p> <p>set_field_init() assigns an application-defined function to be called when the <i>form</i> is posted and just after the current field changes. field_init() returns a pointer to the function, if any.</p> <p>set_field_term() assigns an application-defined function to be called when the <i>form</i> is unposted and just before the current field changes. field_term() returns a pointer to the function, if any.</p> | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <table border="0" style="width: 100%;"> <tr> <td style="padding-right: 40px;">E_OK</td> <td>The function returned successfully.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> </table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. |
| E_OK | The function returned successfully. | | | | |
| E_SYSTEM_ERROR | System error. | | | | |

form_hook(3CURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `courses(3CURSES)`, `forms(3CURSES)`, `attributes(5)`

NOTES The header `<form.h>` automatically includes the headers `<eti.h>` and `<courses.h>`.

| | | | | | |
|----------------------|---|------|-------------------------------------|----------------|---------------|
| NAME | form_hook, set_form_init, form_init, set_form_term, form_term, set_field_init, field_init, set_field_term, field_term – assign application-specific routines for invocation by forms | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_form_init(FORM *form, void (*func) (FORM*)); void (*form_init) (FORM *form); int set_form_term(FORM *form, void (*func) (FORM*)); void (*form_term) (FORM *form); int set_field_init(FORM *form, void (*func) (FORM*)); void (*field_init) (FORM *form); int set_field_term(FORM *form, void (*func) (FORM*)); void (*field_term) (FORM *form);</pre> | | | | |
| DESCRIPTION | <p>These routines allow the programmer to assign application specific routines to be executed automatically at initialization and termination points in the forms application. The user need not specify any application-defined initialization or termination routines at all, but they may be helpful for displaying messages or page numbers and other chores.</p> <p>set_form_init() assigns an application-defined initialization function to be called when the <i>form</i> is posted and just after a page change. form_init() returns a pointer to the initialization function, if any.</p> <p>set_form_term() assigns an application-defined function to be called when the <i>form</i> is unposted and just before a page change. form_term() returns a pointer to the function, if any.</p> <p>set_field_init() assigns an application-defined function to be called when the <i>form</i> is posted and just after the current field changes. field_init() returns a pointer to the function, if any.</p> <p>set_field_term() assigns an application-defined function to be called when the <i>form</i> is unposted and just before the current field changes. field_term() returns a pointer to the function, if any.</p> | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <table border="0" style="width: 100%;"> <tr> <td style="padding-right: 40px;">E_OK</td> <td>The function returned successfully.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> </table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. |
| E_OK | The function returned successfully. | | | | |
| E_SYSTEM_ERROR | System error. | | | | |

form_init(3CURSES)

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), forms(3CURSES), attributes(5)

NOTES The header <form.h> automatically includes the headers <eti.h> and <curses.h>.

| NAME | form_new, new_form, free_form – create and destroy forms | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> FORM *new_form(FIELD **fields) ; int free_form(FORM *form) ;</pre> | | | | |
| DESCRIPTION | <p>new_form() creates a new form connected to the designated fields and returns a pointer to the form.</p> <p>free_form() disconnects the <i>form</i> from its associated field pointer array and deallocates the space for the form.</p> | | | | |
| RETURN VALUES | <p>new_form() always returns NULL on error. free_form() returns one of the following:</p> <p>E_OK The function returned successfully.</p> <p>E_BAD_ARGUMENT An argument is incorrect.</p> <p>E_POSTED The form is posted.</p> | | | | |
| ATTRIBUTES | <p>See attributes(5) for descriptions of the following attributes:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | |

form_new_page(3CURSES)

NAME | form_new_page, set_new_page, new_page – forms pagination

SYNOPSIS |

```
cc [ flag... ] file... -lform -lcurses [ library... ]
#include <form.h>

int set_new_page(FIELD *field, int bool);
int new_page(FIELD *field);
```

DESCRIPTION | set_new_page() marks *field* as the beginning of a new page on the form.
new_page() returns a boolean value indicating whether or not *field* begins a new page of the form.

RETURN VALUES | new_page returns TRUE or FALSE.
set_new_page() returns one of the following:
E_OK The function returned successfully.
E_CONNECTED The field is already connected to a form.
E_SYSTEM_ERROR System error.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5)

NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>.

| NAME | form_opts, set_form_opts, form_opts_on, form_opts_off – forms option routines | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_form_opts(FORM *form, OPTIONS opts); int form_opts_on(FORM *form, OPTIONS opts); int form_opts_off(FORM *form, OPTIONS opts); OPTIONS form_opts(FORM *form);</pre> | | | | |
| DESCRIPTION | <p>set_form_opts() turns on the named options for <i>form</i> and turns off all remaining options. Options are boolean values which can be OR-ed together. form_opts_on() turns on the named options; no other options are changed. form_opts_off() turns off the named options; no other options are changed.</p> <p>form_opts() returns the options set for <i>form</i>.</p> <p>O_NL_OVERLOAD Overload the REQ_NEW_LINE form driver request.</p> <p>O_BS_OVERLOAD Overload the REQ_DEL_PREV form driver request.</p> | | | | |
| RETURN VALUES | <p>set_form_opts(), form_opts_on(), and form_opts_off() return one of the following:</p> <p>E_OK The function returned successfully.</p> <p>E_SYSTEM_ERROR System error.</p> | | | | |
| ATTRIBUTES | See attributes (5) for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | |

form_opts_off(3CURSES)

| NAME | form_opts, set_form_opts, form_opts_on, form_opts_off – forms option routines | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_form_opts(FORM *form, OPTIONS opts); int form_opts_on(FORM *form, OPTIONS opts); int form_opts_off(FORM *form, OPTIONS opts); OPTIONS form_opts(FORM *form);</pre> | | | | |
| DESCRIPTION | <p>set_form_opts() turns on the named options for <i>form</i> and turns off all remaining options. Options are boolean values which can be OR-ed together. form_opts_on() turns on the named options; no other options are changed. form_opts_off() turns off the named options; no other options are changed.</p> <p>form_opts() returns the options set for <i>form</i>.</p> <p>O_NL_OVERLOAD Overload the REQ_NEW_LINE form driver request.</p> <p>O_BS_OVERLOAD Overload the REQ_DEL_PREV form driver request.</p> | | | | |
| RETURN VALUES | <p>set_form_opts(), form_opts_on(), and form_opts_off() return one of the following:</p> <p>E_OK The function returned successfully.</p> <p>E_SYSTEM_ERROR System error.</p> | | | | |
| ATTRIBUTES | See attributes (5) for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | |

| NAME | form_opts, set_form_opts, form_opts_on, form_opts_off – forms option routines | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_form_opts(FORM *form, OPTIONS opts); int form_opts_on(FORM *form, OPTIONS opts); int form_opts_off(FORM *form, OPTIONS opts); OPTIONS form_opts(FORM *form);</pre> | | | | |
| DESCRIPTION | <p>set_form_opts() turns on the named options for <i>form</i> and turns off all remaining options. Options are boolean values which can be OR-ed together. form_opts_on() turns on the named options; no other options are changed. form_opts_off() turns off the named options; no other options are changed.</p> <p>form_opts() returns the options set for <i>form</i>.</p> <p>O_NL_OVERLOAD Overload the REQ_NEW_LINE form driver request.</p> <p>O_BS_OVERLOAD Overload the REQ_DEL_PREV form driver request.</p> | | | | |
| RETURN VALUES | <p>set_form_opts(), form_opts_on(), and form_opts_off() return one of the following:</p> <p>E_OK The function returned successfully.</p> <p>E_SYSTEM_ERROR System error.</p> | | | | |
| ATTRIBUTES | See attributes (5) for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | |

form_page(3CURSES)

NAME | form_page, set_form_page, set_current_field, current_field, field_index – set forms current page and field

SYNOPSIS |

```
cc [ flag... ] file... -lform -lcurses [ library... ]
#include <form.h>

int set_form_page(FORM *form, int page);
int form_page(FORM *form);
int set_current_field(FORM *form, FIELD *field);
FIELD *current_field(FORM*form);
int field_index(FIELD *field);
```

DESCRIPTION | set_form_page() sets the page number of *form* to *page*. form_page() returns the current page number of *form*.

set_current_field() sets the current field of *form* to *field*. current_field() returns a pointer to the current field of *form*.

field_index() returns the index in the field pointer array of *field*.

RETURN VALUES | form_page() returns -1 on error.

current_field() returns NULL on error.

field_index() returns -1 on error.

set_form_page() and set_current_field() return one of the following:

| | |
|------------------|--|
| E_OK | The function returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An argument is incorrect. |
| E_BAD_STATE | The routine was called from an initialization or termination function. |
| E_INVALID_FIELD | The field contents are invalid. |
| E_REQUEST_DENIED | The form driver request failed |

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5)

`form_page(3COURSE)`

NOTES The header `<form.h>` automatically includes the headers `<eti.h>` and `<courses.h>`.

form_post(3CURSES)

| NAME | form_post, post_form, unpost_form – write or erase forms from associated subwindows | | | | | | | | | | | | | | | | |
|----------------------|--|----------------|-------------------------------------|----------------|---------------|----------------|---------------------------|----------|---------------------|--------------|-------------------------|-----------|---|-------------|--|-----------------|---------------------------------------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int post_form(FORM *form) ; int unpost_form(FORM *form) ;</pre> | | | | | | | | | | | | | | | | |
| DESCRIPTION | <p>post_form() writes <i>form</i> into its associated subwindow. The application programmer must use curses library routines to display the form on the physical screen or call update_panels() if the panels library is being used.</p> <p>unpost_form() erases <i>form</i> from its associated subwindow.</p> | | | | | | | | | | | | | | | | |
| RETURN VALUES | <p>These routines return one of the following:</p> <table><tr><td>E_OK</td><td>The function returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr><tr><td>E_BAD_ARGUMENT</td><td>An argument is incorrect.</td></tr><tr><td>E_POSTED</td><td>The form is posted.</td></tr><tr><td>E_NOT_POSTED</td><td>The form is not posted.</td></tr><tr><td>E_NO_ROOM</td><td>The form does not fit in the subwindow.</td></tr><tr><td>E_BAD_STATE</td><td>The routine was called from an initialization or termination function.</td></tr><tr><td>E_NOT_CONNECTED</td><td>The field is not connected to a form.</td></tr></table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An argument is incorrect. | E_POSTED | The form is posted. | E_NOT_POSTED | The form is not posted. | E_NO_ROOM | The form does not fit in the subwindow. | E_BAD_STATE | The routine was called from an initialization or termination function. | E_NOT_CONNECTED | The field is not connected to a form. |
| E_OK | The function returned successfully. | | | | | | | | | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | | | | | | | | | |
| E_BAD_ARGUMENT | An argument is incorrect. | | | | | | | | | | | | | | | | |
| E_POSTED | The form is posted. | | | | | | | | | | | | | | | | |
| E_NOT_POSTED | The form is not posted. | | | | | | | | | | | | | | | | |
| E_NO_ROOM | The form does not fit in the subwindow. | | | | | | | | | | | | | | | | |
| E_BAD_STATE | The routine was called from an initialization or termination function. | | | | | | | | | | | | | | | | |
| E_NOT_CONNECTED | The field is not connected to a form. | | | | | | | | | | | | | | | | |
| ATTRIBUTES | <p>See attributes(5) for descriptions of the following attributes:</p> <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | | | | | | | | | | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | | | | | | | | | | | |
| MT-Level | Unsafe | | | | | | | | | | | | | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), panel_update(3CURSES), panels(3CURSES), attributes(5) | | | | | | | | | | | | | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | | | | | | | | | | | | | |

| NAME | forms – character based forms package | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|--|--------------------|------------------|----------------------------|----------------------------|-------------------------|----------------------------|--------------------------|----------------------------|------------------------|---------------------------------|---------------------------------|----------------------------------|------------------------|--|-------------------------|--|---------------------------|------------------------------------|--------------------------|-----------------------------|-------------------------|--|--------------------------|----------------------------|-------------------------|----------------------------------|-------------------------|----------------------------|
| SYNOPSIS | <code>#include <form.h></code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DESCRIPTION | <p>The <code>form</code> library is built using the <code>curses</code> library, and any program using <code>forms</code> routines must call one of the <code>curses</code> initialization routines such as <code>initscr</code>. A program using these routines must be compiled with <code>-lform</code> and <code>-lcurses</code> on the <code>cc</code> command line.</p> <p>The <code>forms</code> package gives the applications programmer a terminal-independent method of creating and customizing forms for user-interaction. The <code>forms</code> package includes: field routines, which are used to create and customize fields, link fields and assign field types; fieldtype routines, which are used to create new field types for validating fields; and form routines, which are used to create and customize forms, assign pre/post processing functions, and display and interact with forms.</p> | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Current Default Values for Field Attributes | <p>The <code>forms</code> package establishes initial current default values for field attributes. During field initialization, each field attribute is assigned the current default value for that attribute. An application can change or retrieve a current default attribute value by calling the appropriate set or retrieve routine with a <code>NULL</code> field pointer. If an application changes a current default field attribute value, subsequent fields created using <code>new_field()</code> will have the new default attribute value. (The attributes of previously created fields are not changed if a current default attribute value is changed.)</p> | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Routine Name Index | <p>The following table lists each <code>forms</code> routine and the name of the manual page on which it is described.</p> <table border="0"> <thead> <tr> <th style="text-align: left;">forms Routine Name</th> <th style="text-align: left;">Manual Page Name</th> </tr> </thead> <tbody> <tr><td><code>current_field</code></td><td><code>form_page(3X)</code></td></tr> <tr><td><code>data_ahead</code></td><td><code>form_data(3X)</code></td></tr> <tr><td><code>data_behind</code></td><td><code>form_data(3X)</code></td></tr> <tr><td><code>dup_field</code></td><td><code>form_field_new(3X)</code></td></tr> <tr><td><code>dynamic_field_info</code></td><td><code>form_field_info(3X)</code></td></tr> <tr><td><code>field_arg</code></td><td><code>form_field_validation(3X)</code></td></tr> <tr><td><code>field_back</code></td><td><code>form_field_attributes(3X)</code></td></tr> <tr><td><code>field_buffer</code></td><td><code>form_field_buffer(3X)</code></td></tr> <tr><td><code>field_count</code></td><td><code>form_field(3X)</code></td></tr> <tr><td><code>field_fore</code></td><td><code>form_field_attributes(3X)</code></td></tr> <tr><td><code>field_index</code></td><td><code>form_page(3X)</code></td></tr> <tr><td><code>field_info</code></td><td><code>form_field_info(3X)</code></td></tr> <tr><td><code>field_init</code></td><td><code>form_hook(3X)</code></td></tr> </tbody> </table> | forms Routine Name | Manual Page Name | <code>current_field</code> | <code>form_page(3X)</code> | <code>data_ahead</code> | <code>form_data(3X)</code> | <code>data_behind</code> | <code>form_data(3X)</code> | <code>dup_field</code> | <code>form_field_new(3X)</code> | <code>dynamic_field_info</code> | <code>form_field_info(3X)</code> | <code>field_arg</code> | <code>form_field_validation(3X)</code> | <code>field_back</code> | <code>form_field_attributes(3X)</code> | <code>field_buffer</code> | <code>form_field_buffer(3X)</code> | <code>field_count</code> | <code>form_field(3X)</code> | <code>field_fore</code> | <code>form_field_attributes(3X)</code> | <code>field_index</code> | <code>form_page(3X)</code> | <code>field_info</code> | <code>form_field_info(3X)</code> | <code>field_init</code> | <code>form_hook(3X)</code> |
| forms Routine Name | Manual Page Name | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>current_field</code> | <code>form_page(3X)</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>data_ahead</code> | <code>form_data(3X)</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>data_behind</code> | <code>form_data(3X)</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>dup_field</code> | <code>form_field_new(3X)</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>dynamic_field_info</code> | <code>form_field_info(3X)</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>field_arg</code> | <code>form_field_validation(3X)</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>field_back</code> | <code>form_field_attributes(3X)</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>field_buffer</code> | <code>form_field_buffer(3X)</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>field_count</code> | <code>form_field(3X)</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>field_fore</code> | <code>form_field_attributes(3X)</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>field_index</code> | <code>form_page(3X)</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>field_info</code> | <code>form_field_info(3X)</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>field_init</code> | <code>form_hook(3X)</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

forms(3CURSES)

| | |
|-----------------|---------------------------|
| field_just | form_field_just(3X) |
| field_opts | form_field_opts(3X) |
| field_opts_off | form_field_opts(3X) |
| field_opts_on | form_field_opts(3X) |
| field_pad | form_field_attributes(3X) |
| field_status | form_field_buffer(3X) |
| field_term | form_hook(3X) |
| field_type | form_field_validation(3X) |
| field_userptr | form_field_userptr(3X) |
| form_driver | form_driver(3X) |
| form_fields | form_field(3X) |
| form_init | form_hook(3X) |
| form_opts | form_opts(3X) |
| form_opts_off | form_opts(3X) |
| form_opts_on | form_opts(3X) |
| form_page | form_page(3X) |
| form_sub | form_win(3X) |
| form_term | form_hook(3X) |
| form_userptr | form_userptr(3X) |
| form_win | form_win(3X) |
| free_field | form_field_new(3X) |
| free_fieldtype | form_fieldtype(3X) |
| free_form | form_new(3X) |
| link_field | form_field_new(3X) |
| link_fieldtype | form_fieldtype(3X) |
| move_field | form_field(3X) |
| new_field | form_field_new(3X) |
| new_fieldtype | form_fieldtype(3X) |
| new_form | form_new(3X) |
| new_page | form_new_page(3X) |
| pos_form_cursor | form_cursor(3X) |

| | |
|----------------------|---------------------------|
| post_form | form_post(3X) |
| scale_form | form_win(3X) |
| set_current_field | form_page(3X) |
| set_field_back | form_field_attributes(3X) |
| set_field_buffer | form_field_buffer(3X) |
| set_field_fore | form_field_attributes(3X) |
| set_field_init | form_hook(3X) |
| set_field_just | form_field_just(3X) |
| set_field_opts | form_field_opts(3X) |
| set_field_pad | form_field_attributes(3X) |
| set_field_status | form_field_buffer(3X) |
| set_field_term | form_hook(3X) |
| set_field_type | form_field_validation(3X) |
| set_field_userptr | form_field_userptr(3X) |
| set_fieldtype_arg | form_fieldtype(3X) |
| set_fieldtype_choice | form_fieldtype(3X) |
| set_form_fields | form_field(3X) |
| set_form_init | form_hook(3X) |
| set_form_opts | form_opts(3X) |
| set_form_page | form_page(3X) |
| set_form_sub | form_win(3X) |
| set_form_term | form_hook(3X) |
| set_form_userptr | form_userptr(3X) |
| set_form_win | form_win(3X) |
| set_max_field | form_field_buffer(3X) |
| set_new_page | form_new_page(3X) |
| unpost_form | form_post(3X) |

RETURN VALUES

Routines that return a pointer always return `NULL` on error. Routines that return an integer return one of the following:

| | |
|--------------------------|---|
| <code>E_OK</code> | The function returned successfully. |
| <code>E_CONNECTED</code> | The field is already connected to a form. |

forms(3CURSES)

| | |
|-------------------|--|
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An argument is incorrect. |
| E_CURRENT | The field is the current field. |
| E_POSTED | The form is posted. |
| E_NOT_POSTED | The form is not posted. |
| E_INVALID_FIELD | The field contents are invalid. |
| E_NOT_CONNECTED | The field is not connected to a form. |
| E_NO_ROOM | The form does not fit in the subwindow. |
| E_BAD_STATE | The routine was called from an initialization or termination function. |
| E_REQUEST_DENIED | The form driver request failed. |
| E_UNKNOWN_COMMAND | An unknown request was passed to the form driver. |

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3CURSES)`, `attributes(5)`

NOTES The header `<form.h>` automatically includes the headers `<eti.h>` and `<curses.h>`.

| NAME | form_win, set_form_win, set_form_sub, form_sub, scale_form – forms window and subwindow association routines | | | | | | | | | | |
|----------------------|---|----------------|-------------------------------------|----------------|---------------|----------------|---------------------------|-----------------|---------------------------------------|----------|---------------------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_form_win(FORM *form, WINDOW *win); WINDOW *form_win(FORM *form); int set_form_sub(FORM *form, WINDOW *sub); WINDOW *form_sub(FORM *form); int scale_form(FORM *form, int *rows, int *cols);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>set_form_win() sets the window of <i>form</i> to <i>win</i>. form_win() returns a pointer to the window associated with <i>form</i>. set_form_sub() sets the subwindow of <i>form</i> to <i>sub</i>. form_sub() returns a pointer to the subwindow associated with <i>form</i>. scale_form() returns the smallest window size necessary for the subwindow of <i>form</i>. <i>rows</i> and <i>cols</i> are pointers to the locations used to return the number of rows and columns for the form.</p> | | | | | | | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <table border="0"> <tr> <td>E_OK</td> <td>The function returned successfully.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> <tr> <td>E_BAD_ARGUMENT</td> <td>An argument is incorrect.</td> </tr> <tr> <td>E_NOT_CONNECTED</td> <td>The field is not connected to a form.</td> </tr> <tr> <td>E_POSTED</td> <td>The form is posted.</td> </tr> </table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An argument is incorrect. | E_NOT_CONNECTED | The field is not connected to a form. | E_POSTED | The form is posted. |
| E_OK | The function returned successfully. | | | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | | | |
| E_BAD_ARGUMENT | An argument is incorrect. | | | | | | | | | | |
| E_NOT_CONNECTED | The field is not connected to a form. | | | | | | | | | | |
| E_POSTED | The form is posted. | | | | | | | | | | |
| ATTRIBUTES | <p>See attributes(5) for descriptions of the following attributes:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | | | | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | | | | | |
| MT-Level | Unsafe | | | | | | | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | | | | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | | | | | | | |

form_term(3CURSES)

| | | | | | |
|-----------------------------|--|-------------------|-------------------------------------|-----------------------------|---------------|
| NAME | form_hook, set_form_init, form_init, set_form_term, form_term, set_field_init, field_init, set_field_term, field_term – assign application-specific routines for invocation by forms | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_form_init(FORM *form, void (*func)(FORM*)); void (*form_init)(FORM *form); int set_form_term(FORM *form, void (*func)(FORM*)); void (*form_term)(FORM *form); int set_field_init(FORM *form, void (*func)(FORM*)); void (*field_init)(FORM *form); int set_field_term(FORM *form, void (*func)(FORM*)); void (*field_term)(FORM *form);</pre> | | | | |
| DESCRIPTION | <p>These routines allow the programmer to assign application specific routines to be executed automatically at initialization and termination points in the <code>forms</code> application. The user need not specify any application-defined initialization or termination routines at all, but they may be helpful for displaying messages or page numbers and other chores.</p> <p><code>set_form_init()</code> assigns an application-defined initialization function to be called when the <code>form</code> is posted and just after a page change. <code>form_init()</code> returns a pointer to the initialization function, if any.</p> <p><code>set_form_term()</code> assigns an application-defined function to be called when the <code>form</code> is unposted and just before a page change. <code>form_term()</code> returns a pointer to the function, if any.</p> <p><code>set_field_init()</code> assigns an application-defined function to be called when the <code>form</code> is posted and just after the current field changes. <code>field_init()</code> returns a pointer to the function, if any.</p> <p><code>set_field_term()</code> assigns an application-defined function to be called when the <code>form</code> is unposted and just before the current field changes. <code>field_term()</code> returns a pointer to the function, if any.</p> | | | | |
| RETURN VALUES | <p>Routines that return pointers always return <code>NULL</code> on error. Routines that return an integer return one of the following:</p> <table><tr><td><code>E_OK</code></td><td>The function returned successfully.</td></tr><tr><td><code>E_SYSTEM_ERROR</code></td><td>System error.</td></tr></table> | <code>E_OK</code> | The function returned successfully. | <code>E_SYSTEM_ERROR</code> | System error. |
| <code>E_OK</code> | The function returned successfully. | | | | |
| <code>E_SYSTEM_ERROR</code> | System error. | | | | |

form_term(3CURSES)

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), forms(3CURSES), attributes(5)

NOTES The header <form.h> automatically includes the headers <eti.h> and <courses.h>.

form_userptr(3CURSES)

NAME | form_userptr, set_form_userptr – associate application data with forms

SYNOPSIS |

```
cc [ flag... ] file... -lform -lcurses [ library... ]
#include <form.h>

int set_form_userptr(FORM *form, char *ptr);
char *form_userptr(FORM *form);
```

DESCRIPTION | Every form has an associated user pointer that can be used to store pertinent data. set_form_userptr() sets the user pointer of *form*. form_userptr() returns the user pointer of *form*.

RETURN VALUES | form_userptr() returns NULL on error. set_form_userptr() returns one of the following:

| | |
|----------------|-------------------------------------|
| E_OK | The function returned successfully. |
| E_SYSTEM_ERROR | System error. |

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5)

NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>.

| NAME | form_win, set_form_win, set_form_sub, form_sub, scale_form – forms window and subwindow association routines | | | | | | | | | | |
|----------------------|---|----------------|-------------------------------------|----------------|---------------|----------------|---------------------------|-----------------|---------------------------------------|----------|---------------------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_form_win(FORM *form, WINDOW *win); WINDOW *form_win(FORM *form); int set_form_sub(FORM *form, WINDOW *sub); WINDOW *form_sub(FORM *form); int scale_form(FORM *form, int *rows, int *cols);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>set_form_win() sets the window of <i>form</i> to <i>win</i>. form_win() returns a pointer to the window associated with <i>form</i>. set_form_sub() sets the subwindow of <i>form</i> to <i>sub</i>. form_sub() returns a pointer to the subwindow associated with <i>form</i>. scale_form() returns the smallest window size necessary for the subwindow of <i>form</i>. <i>rows</i> and <i>cols</i> are pointers to the locations used to return the number of rows and columns for the form.</p> | | | | | | | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <table border="0"> <tr> <td>E_OK</td> <td>The function returned successfully.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> <tr> <td>E_BAD_ARGUMENT</td> <td>An argument is incorrect.</td> </tr> <tr> <td>E_NOT_CONNECTED</td> <td>The field is not connected to a form.</td> </tr> <tr> <td>E_POSTED</td> <td>The form is posted.</td> </tr> </table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An argument is incorrect. | E_NOT_CONNECTED | The field is not connected to a form. | E_POSTED | The form is posted. |
| E_OK | The function returned successfully. | | | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | | | |
| E_BAD_ARGUMENT | An argument is incorrect. | | | | | | | | | | |
| E_NOT_CONNECTED | The field is not connected to a form. | | | | | | | | | | |
| E_POSTED | The form is posted. | | | | | | | | | | |
| ATTRIBUTES | <p>See attributes(5) for descriptions of the following attributes:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | | | | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | | | | | |
| MT-Level | Unsafe | | | | | | | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | | | | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | | | | | | | |

free_field(3CURSES)

NAME form_field_new, new_field, dup_field, link_field, free_field – create and destroy forms fields

SYNOPSIS

```
cc [ flag... ] file... -lform -lcurses [ library... ]
#include <form.h>

FIELD *new_field(int r, int c, int frow, int fcol, int nrow, int ncol);
FIELD *dup_field(FIELD *field, int frow, int fcol);
FIELD *link_field(FIELD *field, int frow, int fcol);
int free_field(FIELD *field);
```

DESCRIPTION new_field() creates a new field with *r* rows and *c* columns, starting at *frow*, *fcol*, in the subwindow of a form. *nrow* is the number of off-screen rows and *nbuf* is the number of additional working buffers. This routine returns a pointer to the new field.

dup_field() duplicates *field* at the specified location. All field attributes are duplicated, including the current contents of the field buffers.

link_field() also duplicates *field* at the specified location. However, unlike dup_field(), the new field shares the field buffers with the original field. After creation, the attributes of the new field can be changed without affecting the original field.

free_field() frees the storage allocated for *field*.

RETURN VALUES Routines that return pointers return NULL on error. free_field() returns one of the following:

| | |
|----------------|---|
| E_OK | The function returned successfully. |
| E_CONNECTED | The field is already connected to a form. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An argument is incorrect. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), forms(3CURSES), attributes(5)

NOTES The header <form.h> automatically includes the headers <eti.h> and <curses.h>.

free_fieldtype(3CURSES)

| | | | | | | | | | |
|----------------------|--|------|-------------------------------------|----------------|---------------|----------------|---------------------------|-------------|--|
| NAME | form_fieldtype, new_fieldtype, free_fieldtype, set_fieldtype_arg, set_fieldtype_choice, link_fieldtype – forms fieldtype routines | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> FIELDTYPE *new_fieldtype(int (* field_check)(FIELD *, char *), int (*char_check)(int, char *)); int free_fieldtype(FIELDTYPE *fieldtype); int set_fieldtype_arg(FIELDTYPE *fieldtype, char *(* mak_arg)(va_list *), char *(* copy_arg)(char *), void (* free_arg)(char *)); int set_fieldtype_choice(FIELDTYPE *fieldtype, int (* next_choice)(FIELD *, char *), int (*prev_choice)(FIELD *, char *)); FIELDTYPE *link_fieldtype(FIELDTYPE *type1, FIELDTYPE *type2);</pre> | | | | | | | | |
| DESCRIPTION | <p>new_fieldtype() creates a new field type. The application programmer must write the function <i>field_check</i>, which validates the field value, and the function <i>char_check</i>, which validates each character. free_fieldtype() frees the space allocated for the field type.</p> <p>By associating function pointers with a field type, set_fieldtype_arg() connects to the field type additional arguments necessary for a set_field_type() call. Function <i>mak_arg</i> allocates a structure for the field specific parameters to set_field_type() and returns a pointer to the saved data. Function <i>copy_arg</i> duplicates the structure created by <i>make_arg</i>. Function <i>free_arg</i> frees any storage allocated by <i>make_arg</i> or <i>copy_arg</i>.</p> <p>The form_driver() requests REQ_NEXT_CHOICE and REQ_PREV_CHOICE let the user request the next or previous value of a field type comprising an ordered set of values. set_fieldtype_choice() allows the application programmer to implement these requests for the given field type. It associates with the given field type those application-defined functions that return pointers to the next or previous choice for the field.</p> <p>link_fieldtype() returns a pointer to the field type built from the two given types. The constituent types may be any application-defined or pre-defined types.</p> | | | | | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <table><tr><td>E_OK</td><td>The function returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr><tr><td>E_BAD_ARGUMENT</td><td>An argument is incorrect.</td></tr><tr><td>E_CONNECTED</td><td>Type is connected to one or more fields.</td></tr></table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An argument is incorrect. | E_CONNECTED | Type is connected to one or more fields. |
| E_OK | The function returned successfully. | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | |
| E_BAD_ARGUMENT | An argument is incorrect. | | | | | | | | |
| E_CONNECTED | Type is connected to one or more fields. | | | | | | | | |

free_fieldtype(3CURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `courses(3CURSES)`, `forms(3CURSES)`, `attributes(5)`

NOTES The header `<form.h>` automatically includes the headers `<eti.h>` and `< curses.h>`.

| NAME | form_new, new_form, free_form – create and destroy forms | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> FORM *new_form(FIELD **fields) ; int free_form(FORM *form) ;</pre> | | | | |
| DESCRIPTION | <p>new_form() creates a new form connected to the designated fields and returns a pointer to the form.</p> <p>free_form() disconnects the <i>form</i> from its associated field pointer array and deallocates the space for the form.</p> | | | | |
| RETURN VALUES | <p>new_form() always returns NULL on error. free_form() returns one of the following:</p> <p>E_OK The function returned successfully.</p> <p>E_BAD_ARGUMENT An argument is incorrect.</p> <p>E_POSTED The form is posted.</p> | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | |

free_item(3CURSES)

NAME menu_item_new, new_item, free_item – create and destroy menu items

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

ITEM *new_item(char *name, char *desc);
int free_item(ITEM *item);
```

DESCRIPTION new_item() creates a new item from *name* and *description*, and returns a pointer to the new item.

free_item() frees the storage allocated for *item*. Once an item is freed, the user can no longer connect it to a menu.

RETURN VALUES new_item() returns NULL on error.

free_item() returns one of the following:

| | |
|----------------|--|
| E_OK | The routine returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. |
| E_CONNECTED | One or more items are already connected to another menu. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), attributes(5)

NOTES The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

free_menu(3CURSES)

NAME menu_new, new_menu, free_menu – create and destroy menus

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

MENU *new_menu (ITEM **items) ;
int free_menu (MENU *menu) ;
```

DESCRIPTION new_menu () creates a new menu connected to the item pointer array *items* and returns a pointer to the new menu.

free_menu () disconnects *menu* from its associated item pointer array and frees the storage allocated for the menu.

RETURN VALUES new_menu () returns NULL on error.

free_menu () returns one of the following:

E_OK The routine returned successfully.

E_SYSTEM_ERROR System error.

E_BAD_ARGUMENT An incorrect argument was passed to the routine.

E_POSTED The menu is already posted.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), attributes(5)

NOTES The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

getbegyx(3CURSES)

| NAME | <code> curs_getyx, getyx, getparyx, getbegyx, getmaxyx</code> – get curses cursor and window coordinates | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> void getyx(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>); void getparyx(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>); void getbegyx(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>); void getmaxyx(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>);</pre> | | | | |
| DESCRIPTION | <p>With the <code>getyx()</code> macro, the cursor position of the window is placed in the two integer variables <i>y</i> and <i>x</i>.</p> <p>With the <code>getparyx()</code> macro, if <i>win</i> is a subwindow, the beginning coordinates of the subwindow relative to the parent window are placed into two integer variables, <i>y</i> and <i>x</i>. Otherwise, <code>-1</code> is placed into <i>y</i> and <i>x</i>.</p> <p>Like <code>getyx()</code>, the <code>getbegyx()</code> and <code>getmaxyx()</code> macros store the current beginning coordinates and size of the specified window.</p> | | | | |
| RETURN VALUES | The return values of these macros are undefined (that is, they should not be used as the right-hand side of assignment statements). | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that all of these interfaces are macros and that “&” is not necessary before the variables <i>y</i> and <i>x</i>.</p> | | | | |

| NAME | getbegyx, getmaxyx, getparyx, getyx – get cursor or window coordinates | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> void getbegyx(WINDOW *win, int y, int x); void getmaxyx(WINDOW *win, int y, int x); void getparyx(WINDOW *win, int y, int x); void getyx(WINDOW *win, int y, int x);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>getyx()</code> macro stores the current cursor position of the specified window in <i>x</i> and <i>y</i>.</p> <p>The <code>getparyx()</code> macro stores the <i>x</i> and <i>y</i> coordinates (relative to the parent window) of the specified window's origin (upper-left corner). If <i>win</i> does not point to a subwindow, <i>x</i> and <i>y</i> are set to -1.</p> <p>The <code>getbegyx()</code> macro stores the <i>x</i> and <i>y</i> coordinates of the specified window's origin (upper-left corner).</p> <p>The <code>getmaxyx()</code> macro stores the numbers of rows in the specified window in <i>y</i> and the number of columns in <i>x</i>.</p> | | | | | | |
| PARAMETERS | <p><i>win</i> Is a pointer to a window.</p> <p><i>y</i> stores the <i>y</i> coordinate for the cursor or origin. The <code>getmaxyx()</code> macro uses it to store the number of rows in the window.</p> <p><i>x</i> stores the <i>x</i> coordinate for the cursor or origin. The <code>getmaxyx()</code> macro uses it to store the number of columns in the window.</p> | | | | | | |
| RETURN VALUES | These macros do not return a value. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

getbkgd(3XCURSES)

| | |
|----------------------|---|
| NAME | <code>bkgd</code> , <code>bkgdset</code> , <code>getbkgd</code> , <code>wbkgd</code> , <code>wbkgdset</code> – set or get the background character (and rendition) of window |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int bkgd(chtype <i>ch</i>); void bkgdset(chtype <i>ch</i>); chtype getbkgd(WINDOW *<i>win</i>); int wbkgd(WINDOW *<i>win</i>, chtype <i>ch</i>); void wbkgdset(WINDOW *<i>win</i>, chtype <i>ch</i>);</pre> |
| DESCRIPTION | <p>The <code>bkgdset()</code> and <code>wbkgdset()</code> functions turn off the previous background attributes, logical OR the requested attributes into the window rendition, and set the background property of the current or specified window based on the information in <i>ch</i>. If <i>ch</i> refers to a multi-column character, the results are undefined.</p> <p>The <code>bkgd()</code> and <code>wbkgd()</code> functions turn off the previous background attributes, logical OR the requested attributes into the window rendition, and set the background property of the current or specified window and then apply this setting to every character position in that window:</p> <ul style="list-style-type: none">■ The rendition of every character on the screen is changed to the new window rendition.■ Wherever the former background character appears, it is changed to the new background character. <p>The <code>getbkgd()</code> function extracts the specified window's background character and rendition.</p> |
| PARAMETERS | <p><i>ch</i> Is the background character to be set.</p> <p><i>win</i> Is a pointer to the window in which the background character is to be set.</p> |
| RETURN VALUES | <p>Upon successful completion, the <code>bkgd()</code> and <code>wbkgd()</code> functions return OK. Otherwise, they return ERR.</p> <p>The <code>bkgdset()</code> and <code>wbkgdset()</code> functions do not return a value.</p> <p>Upon successful completion, the <code>getbkgd()</code> function returns the specified window's background character and rendition. Otherwise, it returns (chtype) ERR.</p> |
| ERRORS | No errors are defined. |

getbkgd(3XCURSES)

USAGE These functions are only guaranteed to operate reliably on character sets in which each character fits into a single byte, whose attributes can be expressed using only constants with the A_ prefix.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO addch(3XCURSES), addchstr(3XCURSES), attroff(3XCURSES), bkgrnd(3XCURSES), clear(3XCURSES), clrtoeol(3XCURSES), clrtoeol(3XCURSES), clrtobot(3XCURSES), erase(3XCURSES), inch(3XCURSES), libcurses(3XCURSES), mvprintw(3XCURSES), attributes(5), standards(5)

getbkgrnd(3XCURSES)

| | |
|----------------------|--|
| NAME | <code>bkgrnd</code> , <code>bkgrndset</code> , <code>getbkgrnd</code> , <code>wbkgrnd</code> , <code>wbkgrndset</code> , <code>wgetbkgrnd</code> – set or get the background character (and rendition) of window using a complex character |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int bkgrnd(const cchar_t *wch); void bkgrndset(const cchar_t *wch); int getbkgrnd(cchar_t *wch); int wbkgrnd(WINDOW *win, const cchar_t *wch); void wbkgrndset(WINDOW *win, const cchar_t *wch); int wgetbkgrnd(WINDOW *win, cchar_t *wch);</pre> |
| DESCRIPTION | <p>The <code>bkgrndset()</code> and <code>wbkgrndset()</code> functions turn off the previous background attributes, logical OR the requested attributes into the window rendition, and set the background property of the current or specified window based on the information in <i>wch</i>.</p> <p>The <code>bkgrnd()</code> and <code>wbkgrnd()</code> functions turn off the previous background attributes, logical OR the requested attributes into the window rendition, and set the background property of the current or specified window and then apply this setting to every character position in that window:</p> <ul style="list-style-type: none">■ The rendition of every character on the screen is changed to the new window rendition.■ Wherever the former background character appears, it is changed to the new background character. <p>If <i>wch</i> refers to a non-spacing complex character for <code>bkgrnd()</code>, <code>bkgrndset()</code>, <code>wbkgrnd()</code>, and <code>wbkgrndset()</code>, then <i>wch</i> is added to the existing spacing complex character that is the background character. If <i>wch</i> refers to a multi-column character, the results are unspecified.</p> <p>The <code>getbkgrnd()</code> and <code>wgetbkgrnd()</code> functions store, into the area pointed to by <i>wch</i>, the window's background character and rendition.</p> |
| PARAMETERS | <p><i>wch</i> Is a pointer to the complex background character to be set.</p> <p><i>win</i> Is a pointer to the window in which the complex background character is to be set.</p> |
| RETURN VALUES | <p>The <code>bkgrndset()</code> and <code>wbkgrndset()</code> functions do not return a value.</p> <p>Upon successful completion, the other functions return OK. Otherwise, they return ERR.</p> |

getbkgrnd(3XCURSES)

ERRORS No errors are defined.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `add_wch(3XCURSES)`, `add_wchnstr(3XCURSES)`, `addch(3XCURSES)`, `addchstr(3XCURSES)`, `attroff(3XCURSES)`, `bkgd(3XCURSES)`, `clear(3XCURSES)`, `clrtoeol(3XCURSES)`, `clrtobot(3XCURSES)`, `erase(3XCURSES)`, `inch(3XCURSES)`, `libcurses(3XCURSES)`, `mvprintw(3XCURSES)`, `attributes(5)`, `standards(5)`

getcchar(3XCURSES)

NAME | getcchar – get a wide character string (with rendition) from a cchar_t

SYNOPSIS | `cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \`
`-R /usr/xpg4/lib -lcurses [library...]`

c89 | `[flag...] file... -lcurses [library...]`

`#include <curses.h>`

`int getcchar(const cchar_t *wcv, wchar_t *wch, attr_t *attrs, short`
`*color_pair, void *opt);`

DESCRIPTION | If *wch* is not a null pointer, the `getcchar()` function splits the `cchar_t` object pointed to by *wcv* into a wide character string, attributes, and a color pair. It stores the attributes in the location pointed to by *attrs*, the color pair in the location pointed to by *color_pair*, and the wide character string in the location pointed to by *wch*.

If *wch* is a null pointer, the `getcchar()` function simply returns the number of wide characters in the `cchar_t` object pointed to by *wcv*. The objects pointed to by *attrs* and *color_pair* are not changed.

PARAMETERS | *wcv* | Is a pointer to a `cchar_t` object.

wch | Is a pointer to an object where a wide character string can be stored.

attrs | Is a pointer to an object where attributes can be stored.

color_pair | Is a pointer to an object where a color pair can be stored.

opt | Is reserved for future use. Currently, this must be a null pointer.

RETURN VALUES | When *wch* is a null pointer, the `getcchar()` function returns the number of wide characters in the string pointed to by *wcv* including the null terminator.

When *wch* is not a null pointer, the `getcchar()` function returns OK on success and ERR otherwise.

ERRORS | None

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | `attroff(3XCURSES)`, `can_change_color(3XCURSES)`, `libcurses(3XCURSES)`, `setcchar(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_getch, getch, wgetch, mvgetch, mvwgetch, ungetch – get (or push back) characters from curses terminal keyboard | | | | |
|----------------------|--|------|----------|-----------|-----------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int getch(void); int wgetch(WINDOW *win); int mvgetch(int y, int x); int mvwgetch(WINDOW *win, int y, int x); int ungetch(int ch);</pre> | | | | |
| DESCRIPTION | <p>With the <code>getch()</code>, <code>wgetch()</code>, <code>mvgetch()</code>, and <code>mvwgetch()</code> routines a character is read from the terminal associated with the window. In no-delay mode, if no input is waiting, the value <code>ERR</code> is returned. In delay mode, the program waits until the system passes text through to the program. Depending on the setting of <code>cbreak()</code>, this is after one character (<code>cbreak</code> mode), or after the first newline (<code>nocbreak</code> mode). In half-delay mode, the program waits until a character is typed or the specified timeout has been reached. Unless <code>noecho()</code> has been set, the character will also be echoed into the designated window.</p> <p>If the window is not a pad, and it has been moved or modified since the last call to <code>wrefresh()</code>, <code>wrefresh()</code> will be called before another character is read.</p> <p>If <code>keypad()</code> is <code>TRUE</code>, and a function key is pressed, the token for that function key is returned instead of the raw characters. Possible function keys are defined in <code><curses.h></code> with integers beginning with <code>0401</code>, whose names begin with <code>KEY_</code>. If a character that could be the beginning of a function key (such as escape) is received, <code>curses</code> sets a timer. If the remainder of the sequence does not come in within the designated time, the character is passed through; otherwise, the function key value is returned. For this reason, many terminals experience a delay between the time a user presses the escape key and the escape is returned to the program. Since tokens returned by these routines are outside the ASCII range, they are not printable.</p> <p>The <code>ungetch()</code> routine places <code>ch</code> back onto the input queue to be returned by the next call to <code>wgetch()</code>.</p> | | | | |
| Function Keys | <p>The following function keys, defined in <code><curses.h></code>, might be returned by <code>getch()</code> if <code>keypad()</code> has been enabled. Note that not all of these may be supported on a particular terminal if the terminal does not transmit a unique code when the key is pressed or if the definition for the key is not present in the <code>terminfo</code> database.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Name</th> <th style="text-align: left; padding: 2px;">Key name</th> </tr> </thead> <tbody> <tr> <td style="padding: 2px;">KEY_BREAK</td> <td style="padding: 2px;">Break key</td> </tr> </tbody> </table> | Name | Key name | KEY_BREAK | Break key |
| Name | Key name | | | | |
| KEY_BREAK | Break key | | | | |

getch(3CURSES)

| <i>Name</i> | <i>Key name</i> |
|-------------------|---|
| KEY_DOWN | The four arrow keys . . . |
| KEY_UP | |
| KEY_LEFT | |
| KEY_RIGHT | |
| KEY_HOME | Home key (upward+left arrow) |
| KEY_BACKSPACE | Backspace |
| KEY_F0 | Function keys; space for 64 keys is reserved. |
| KEY_F(<i>n</i>) | For $0 \leq n \leq 63$ |
| KEY_DL | Delete line |
| KEY_IL | Insert line |
| KEY_DC | Delete character |
| KEY_IC | Insert char or enter insert mode |
| KEY_EIC | Exit insert char mode |
| KEY_CLEAR | Clear screen |
| KEY_EOS | Clear to end of screen |
| KEY_EOL | Clear to end of line |
| KEY_SF | Scroll 1 line forward |
| KEY_SR | Scroll 1 line backward (reverse) |
| KEY_NPAGE | Next page |
| KEY_PPAGE | Previous page |
| KEY_STAB | Set tab |
| KEY_CTAB | Clear tab |
| KEY_CATAB | Clear all tabs |
| KEY_ENTER | Enter or send |
| KEY_SRESET | Soft (partial) reset |
| KEY_RESET | Reset or hard reset |
| KEY_PRINT | Print or copy |
| KEY_LL | Home down or bottom (lower left). Keypad is arranged like this: (Row 1) A1 up A3 (Row 2) left B2 right (Row 3) C1 down C3 |

| <i>Name</i> | <i>Key name</i> |
|---------------|-----------------------|
| KEY_A1 | Upper left of keypad |
| KEY_A3 | Upper right of keypad |
| KEY_B2 | Center of keypad |
| KEY_C1 | Lower left of keypad |
| KEY_C3 | Lower right of keypad |
| KEY_BTAB | Back tab key |
| KEY_BEG | Beg(inning) key |
| KEY_CANCEL | Cancel key |
| KEY_CLOSE | Close key |
| KEY_COMMAND | Cmd (command) key |
| KEY_COPY | Copy key |
| KEY_CREATE | Create key |
| KEY_END | End key |
| KEY_EXIT | Exit key |
| KEY_FIND | Find key |
| KEY_HELP | Help key |
| KEY_MARK | Mark key |
| KEY_MESSAGE | Message key |
| KEY_MOVE | Move key |
| KEY_NEXT | Next object key |
| KEY_OPEN | Open key |
| KEY_OPTIONS | Options key |
| KEY_PREVIOUS | Previous object key |
| KEY_REDO | Redo key |
| KEY_REFERENCE | Reference key |
| KEY_REFRESH | Refresh key |
| KEY_REPLACE | Replace key |
| KEY_RESTART | Restart key |
| KEY_RESUME | Resume key |

getch(3CURSES)

| <i>Name</i> | <i>Key name</i> |
|---------------|-------------------------|
| KEY_SAVE | Save key |
| KEY_SBEG | Shifted beginning key |
| KEY_SCANCEL | Shifted cancel key |
| KEY_SCOMMAND | Shifted command key |
| KEY_SCOPY | Shifted copy key |
| KEY_SCREATE | Shifted create key |
| KEY_SDC | Shifted delete char key |
| KEY_SDL | Shifted delete line key |
| KEY_SELECT | Select key |
| KEY_SEND | Shifted end key |
| KEY_SEOL | Shifted clear line key |
| KEY_SEXIT | Shifted exit key |
| KEY_SFIND | Shifted find key |
| KEY_SHELP | Shifted help key |
| KEY_SHOME | Shifted home key |
| KEY_SIC | Shifted input key |
| KEY_SLEFT | Shifted left arrow key |
| KEY_SMESSAGE | Shifted message key |
| KEY_SMOVE | Shifted move key |
| KEY_SNEXT | Shifted next key |
| KEY_SOPTIONS | Shifted options key |
| KEY_SPREVIOUS | Shifted prev key |
| KEY_SPRINT | Shifted print key |
| KEY_SREDO | Shifted redo key |
| KEY_SREPLACE | Shifted replace key |
| KEY_SRIGHT | Shifted right arrow |
| KEY_SRSUME | Shifted resume key |
| KEY_SSAVE | Shifted save key |
| KEY_SSUSPEND | Shifted suspend key |

| <i>Name</i> | <i>Key name</i> |
|-------------|------------------|
| KEY_SUNDO | Shifted undo key |
| KEY_SUSPEND | Suspend key |
| KEY_UNDO | Undo key |

RETURN VALUES All routines return the integer `ERR` upon failure. The `ungetch()` routine returns an integer value other than `ERR` upon successful completion. The other routines return the next input character or function key code upon successful completion.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_inopts(3CURSES)`, `curs_move(3CURSES)`, `curs_refresh(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Use of the escape key for a single character function is discouraged.

When using `getch()`, `wgetch()`, `mvgetch()`, or `mvwgetch()`, `nocbreak` mode (`nocbreak()`) and `echo` mode (`echo()`) should not be used at the same time. Depending on the state of the tty driver when each character is typed, the program may produce undesirable results.

Note that `getch()`, `mvgetch()`, and `mvwgetch()` may be macros.

getch(3XCURSES)

| | | | | | |
|--|--|--------------------------------|---|---------------|--|
| NAME | getch, wgetch, mvgetch, mvwgetch – get a single-byte character from the terminal | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int getch(void); int wgetch(WINDOW *win); int mvgetch(int y, int x); int mvwgetch(WINDOW *win, int y, int x);</pre> | | | | |
| PARAMETERS | <p><i>win</i> Is a pointer to the window associated with the terminal from which the character is to be read.</p> <p><i>y</i> Is the y (row) coordinate for the position of the character to be read.</p> <p><i>x</i> Is the x (column) coordinate for the position of the character to be read.</p> | | | | |
| DESCRIPTION | <p>These functions read a single-byte character from the terminal associated with the current or specified window. The results are unspecified if the input is not a single-byte character. If keypad(3XCURSES) is enabled, these functions respond to the pressing of a function key by returning the corresponding <code>KEY_</code> value defined in <code><curses.h></code></p> <p>Processing of terminal input is subject to the general rules described on the keypad(3XCURSES) manual page.</p> <p>If echoing is enabled, then the character is echoed as though it were provided as an input argument to <code>addch(3XCURSES)</code>, except for the following characters:</p> <table><tr><td><code><backspace></code></td><td>The input is interpreted as follows: unless the cursor already was in column 0, <code><backspace></code> moves the cursor one column toward the start of the current line and any characters after the <code><backspace></code> are added or inserted starting there. The character at the resulting cursor position is then deleted as though <code>delch(3XCURSES)</code> were called, except that if the cursor was originally in the first column of the line, the user is alerted as though <code>beep(3XCURSES)</code> were called.</td></tr><tr><td>Function keys</td><td>The user is alerted as though <code>beep()</code> were called. Information concerning the function keys is not returned to the caller.</td></tr></table> <p>If the current or specified window is not a pad, and it has been moved modified since the last refresh operation, then it will be refreshed before another character is read.</p> | <code><backspace></code> | The input is interpreted as follows: unless the cursor already was in column 0, <code><backspace></code> moves the cursor one column toward the start of the current line and any characters after the <code><backspace></code> are added or inserted starting there. The character at the resulting cursor position is then deleted as though <code>delch(3XCURSES)</code> were called, except that if the cursor was originally in the first column of the line, the user is alerted as though <code>beep(3XCURSES)</code> were called. | Function keys | The user is alerted as though <code>beep()</code> were called. Information concerning the function keys is not returned to the caller. |
| <code><backspace></code> | The input is interpreted as follows: unless the cursor already was in column 0, <code><backspace></code> moves the cursor one column toward the start of the current line and any characters after the <code><backspace></code> are added or inserted starting there. The character at the resulting cursor position is then deleted as though <code>delch(3XCURSES)</code> were called, except that if the cursor was originally in the first column of the line, the user is alerted as though <code>beep(3XCURSES)</code> were called. | | | | |
| Function keys | The user is alerted as though <code>beep()</code> were called. Information concerning the function keys is not returned to the caller. | | | | |
| Constant Values for Function Keys | The following is a list of tokens for function keys that are returned by the <code>getch()</code> set of functions if keypad handling is enabled (some terminals may not support all tokens). | | | | |

| Constant | Description |
|-------------------|---|
| KEY_BREAK | Break key |
| KEY_DOWN | The down arrow key |
| KEY_UP | The up arrow key |
| KEY_LEFT | The left arrow key |
| KEY_RIGHT | The right arrow key |
| KEY_HOME | Home key |
| KEY_BACKSPACE | Backspace |
| KEY_F0 | Function keys. Space for 64 keys is reserved. |
| KEY_F(<i>n</i>) | For $0 \leq n \leq 63$ |
| KEY_DL | Delete line |
| KEY_IL | Insert line |
| KEY_DC | Delete character |
| KEY_IC | Insert char or enter insert mode |
| KEY_EIC | Exit insert char mode |
| KEY_CLEAR | Clear screen |
| KEY_EOS | Clear to end of screen |
| KEY_EOL | Clear to end of line |
| KEY_SF | Scroll 1 line forward |
| KEY_SR | Scroll 1 line backwards |
| KEY_NPAGE | Next page |
| KEY_PPAGE | Previous page |
| KEY_STAB | Set tab |
| KEY_CTAB | Clear tab |
| KEY_CATAB | Clear all tabs |
| KEY_ENTER | Enter or send |
| KEY_SRESET | Soft (partial) reset |
| KEY_RESET | Reset or hard reset |
| KEY_PRINT | Print or copy |
| KEY_LL | Home down or bottom (lower left) |

getch(3XCURSES)

| Constant | Description |
|---------------|-----------------------|
| KEY_A1 | Upper left of keypad |
| KEY_A3 | Upper right of keypad |
| KEY_B2 | Center of keypad |
| KEY_C1 | Lower left of keypad |
| KEY_C3 | Lower right of keypad |
| KEY_BTAB | Back tab |
| KEY_BEG | Beginning key |
| KEY_CANCEL | Cancel key |
| KEY_CLOSE | Close key |
| KEY_COMMAND | Cmd (command) key |
| KEY_COPY | Copy key |
| KEY_CREATE | Create key |
| KEY_END | End key |
| KEY_EXIT | Exit key |
| KEY_FIND | Find key |
| KEY_HELP | Help key |
| KEY_MARK | Mark key |
| KEY_MESSAGE | Message key |
| KEY_MOVE | Move key |
| KEY_NEXT | Next object key |
| KEY_OPEN | Open key |
| KEY_OPTIONS | Options key |
| KEY_PREVIOUS | Previous object key |
| KEY_REDO | Redo key |
| KEY_REFERENCE | Reference key |
| KEY_REFRESH | Refresh key |
| KEY_REPLACE | Replace key |
| KEY_RESTART | Restart key |
| KEY_RESUME | Resume key |

| Constant | Description |
|---------------|-------------------------|
| KEY_SAVE | Save key |
| KEY_SBEG | Shifted beginning key |
| KEY_SCANCEL | Shifted cancel key |
| KEY_SCOMMAND | Shifted command key |
| KEY_SCOPY | Shifted copy key |
| KEY_SCREATE | Shifted create key |
| KEY_SDC | Shifted delete char key |
| KEY_SDL | Shifted delete line key |
| KEY_SELECT | Select key |
| KEY_SEND | Shifted end key |
| KEY_SEOL | Shifted clear line key |
| KEY_SEXIT | Shifted exit key |
| KEY_SFIND | Shifted find key |
| KEY_SHELP | Shifted help key |
| KEY_SHOME | Shifted home key |
| KEY_SIC | Shifted input key |
| KEY_SLEFT | Shifted left arrow key |
| KEY_SMESSAGES | Shifted messages key |
| KEY_SMOVE | Shifted move key |
| KEY_SNEXT | Shifted next key |
| KEY_SOPTIONS | Shifted options key |
| KEY_SPREVIOUS | Shifted previous key |
| KEY_SPRINT | Shifted print key |
| KEY_SREDO | Shifted redo key |
| KEY_SREPLACE | Shifted replace key |
| KEY_SRIGHT | Shifted right arrow key |
| KEY_SRSUME | Shifted resume key |
| KEY_SSAVE | Shifted save key |
| KEY_SSUSPEND | Shifted suspend key |

getch(3XCURSES)

| Constant | Description |
|-------------|------------------|
| KEY_SUNDO | Shifted undo key |
| KEY_SUSPEND | Suspend key |
| KEY_UNDO | Undo key |

RETURN VALUES Upon successful completion, these functions return the single-byte character, `KEY_` value, or `ERR`. When in the nodelay mode and no data is available, `ERR` is returned.

ERRORS No errors are defined.

USAGE Applications should not define the escape key by itself as a single-character function. When using these functions, `nocbreak` mode (`cbreak(3XCURSES)`) and `echo` mode (`echo(3XCURSES)`) should not be used at the same time. Depending on the state of the terminal when each character is typed, the application may produce undesirable results.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `cbreak(3XCURSES)`, `echo(3XCURSES)`, `halfdelay(3XCURSES)`, `keypad(3XCURSES)`, `libcurses(3XCURSES)`, `nodelay(3XCURSES)`, `notimeout(3XCURSES)`, `raw(3XCURSES)`, `timeout(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_getyx, getyx, getparyx, getbegyx, getmaxyx – get curses cursor and window coordinates | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> void getyx(WINDOW *win, int y, int x); void getparyx(WINDOW *win, int y, int x); void getbegyx(WINDOW *win, int y, int x); void getmaxyx(WINDOW *win, int y, int x);</pre> | | | | |
| DESCRIPTION | <p>With the <code>getyx()</code> macro, the cursor position of the window is placed in the two integer variables <code>y</code> and <code>x</code>.</p> <p>With the <code>getparyx()</code> macro, if <code>win</code> is a subwindow, the beginning coordinates of the subwindow relative to the parent window are placed into two integer variables, <code>y</code> and <code>x</code>. Otherwise, <code>-1</code> is placed into <code>y</code> and <code>x</code>.</p> <p>Like <code>getyx()</code>, the <code>getbegyx()</code> and <code>getmaxyx()</code> macros store the current beginning coordinates and size of the specified window.</p> | | | | |
| RETURN VALUES | The return values of these macros are undefined (that is, they should not be used as the right-hand side of assignment statements). | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that all of these interfaces are macros and that “&” is not necessary before the variables <code>y</code> and <code>x</code>.</p> | | | | |

getmaxyx(3XCURSES)

| NAME | getbegyx, getmaxyx, getparyx, getyx – get cursor or window coordinates | | | | | | |
|----------------------|--|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> void getbegyx(WINDOW *win, int y, int x); void getmaxyx(WINDOW *win, int y, int x); void getparyx(WINDOW *win, int y, int x); void getyx(WINDOW *win, int y, int x);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>getyx()</code> macro stores the current cursor position of the specified window in <i>x</i> and <i>y</i>.</p> <p>The <code>getparyx()</code> macro stores the <i>x</i> and <i>y</i> coordinates (relative to the parent window) of the specified window's origin (upper-left corner). If <i>win</i> does not point to a subwindow, <i>x</i> and <i>y</i> are set to <code>-1</code>.</p> <p>The <code>getbegyx()</code> macro stores the <i>x</i> and <i>y</i> coordinates of the specified window's origin (upper-left corner).</p> <p>The <code>getmaxyx()</code> macro stores the numbers of rows in the specified window in <i>y</i> and the number of columns in <i>x</i>.</p> | | | | | | |
| PARAMETERS | <p><i>win</i> Is a pointer to a window.</p> <p><i>y</i> stores the <i>y</i> coordinate for the cursor or origin. The <code>getmaxyx()</code> macro uses it to store the number of rows in the window.</p> <p><i>x</i> stores the <i>x</i> coordinate for the cursor or origin. The <code>getmaxyx()</code> macro uses it to store the number of columns in the window.</p> | | | | | | |
| RETURN VALUES | These macros do not return a value. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Standard</td></tr><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

| | | | | | | | | | | | |
|--------------------|--|------------|--|----------|---|----------|--|----------|---|------------|---|
| NAME | getnstr, getstr, mvgetnstr, mvgetstr, mvwgetnstr, mvwgetstr, wgetnstr, wgetstr – get a multibyte character string from terminal | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int getnstr(char *str, int n); int getstr(char *str); int mvgetnstr(int y, int x, char *str, int n); int mvgetstr(int y, int x, char *str); int mvwgetnstr(WINDOW *win, int y, int x, char *str, int n); int mvwgetstr(WINDOW *win, int y, int x, char *str); int wgetnstr(WINDOW *win, char *str, int n); int wgetstr(WINDOW *win, char *str);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The <code>getstr()</code> and <code>wgetstr()</code> functions get a character string from the terminal associated with the window <code>stdscr</code> or window <code>win</code>, respectively. The <code>mvgetstr()</code> and <code>mvwgetstr()</code> functions move the cursor to the position specified in <code>stdscr</code> or <code>win</code>, respectively, then get a character string.</p> <p>These functions call <code>wgetch(3XCURSES)</code> and place each received character in <code>str</code> until a newline is received, which is also placed in <code>str</code>. The erase and kill characters set by the user are processed.</p> <p>The <code>getnstr()</code>, <code>mvgetnstr()</code>, <code>mvwgetnstr()</code> and <code>wgetnstr()</code> functions read at most <code>n</code> characters. These functions are used to prevent overflowing the input buffer.</p> <p>The <code>getnstr()</code>, <code>wgetnstr()</code>, <code>mvgetnstr()</code>, and <code>mvwgetnstr()</code> functions only return complete multibyte characters. If the area pointed to by <code>str</code> is not large enough to hold at least one character, these functions fail.</p> | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>str</i></td> <td>Is a pointer to the area where the character string is to be placed.</td> </tr> <tr> <td><i>n</i></td> <td>Is the maximum number of characters to read from input.</td> </tr> <tr> <td><i>y</i></td> <td>Is the y (row) coordinate of starting position of character string to be read.</td> </tr> <tr> <td><i>x</i></td> <td>Is the x (column) coordinate of starting position of character string to be read.</td> </tr> <tr> <td><i>win</i></td> <td>Points to the window associated with the terminal from which the character is to be read.</td> </tr> </table> | <i>str</i> | Is a pointer to the area where the character string is to be placed. | <i>n</i> | Is the maximum number of characters to read from input. | <i>y</i> | Is the y (row) coordinate of starting position of character string to be read. | <i>x</i> | Is the x (column) coordinate of starting position of character string to be read. | <i>win</i> | Points to the window associated with the terminal from which the character is to be read. |
| <i>str</i> | Is a pointer to the area where the character string is to be placed. | | | | | | | | | | |
| <i>n</i> | Is the maximum number of characters to read from input. | | | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of starting position of character string to be read. | | | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of starting position of character string to be read. | | | | | | | | | | |
| <i>win</i> | Points to the window associated with the terminal from which the character is to be read. | | | | | | | | | | |

getnstr(3XCURSES)

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `getch(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_getwstr, getwstr, getnwstr, wgetwstr, wgetnwstr, mvgetwstr, mvgetnwstr, mvwgetwstr, mvwgetnwstr – get wchar_t character strings from curses terminal keyboard | | | | |
|---------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int getwstr(wchar_t *wstr); int getnwstr(wchar_t *wstr, int n); int wgetwstr(WINDOW *win, wchar_t *wstr); int wgetnwstr(WINDOW *win, wchar_t *wstr, int n); int mvgetwstr(int y, int x, wchar_t *wstr); int mvgetnwstr(int y, int x, wchar_t *wstr, int n); int mvwgetwstr(WINDOW *win, int y, int x, wchar_t *wstr); int mvwgetnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);</pre> | | | | |
| DESCRIPTION | The effect of getwstr() is as though a series of calls to getwch(3CURSES) were made, until a newline and carriage return is received. The resulting value is placed in the area pointed to by the wchar_t pointer wstr. getnwstr() reads at most n wchar_t characters, thus preventing a possible overflow of the input buffer. The user's erase and kill characters are interpreted, as well as any special keys (such as function keys, HOME key, CLEAR key, etc.). | | | | |
| RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See attributes(5) for a description of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), getwch(3CURSES), attributes(5) | | | | |
| NOTES | <p>The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h>, and <wctype.h>.</p> <p>Note that all routines except wgetnwstr() may be macros.</p> | | | | |

getn_wstr(3XCURSES)

| | | | | | | | | | | | |
|----------------------|---|-------------|--|----------|---|----------|--|----------|---|------------|---|
| NAME | getn_wstr, get_wstr, mvgetn_wstr, mvget_wstr, mvwgetn_wstr, mvwget_wstr, wgetn_wstr, wget_wstr – get a wide character string from terminal | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int getn_wstr(wint_t *wstr, int n); int get_wstr(wint_t *wstr); int mvgetn_wstr(int y, int x, wint_t *wstr, int n); int mvget_wstr(int y, int x, wint_t *wstr); int mvwgetn_wstr(WINDOW *win, int y, int x, wint_t *wstr, int n); int mvwget_wstr(WINDOW *win, int y, int x, wint_t *wstr); int wgetn_wstr(WINDOW *win, wint_t *wstr, int n); int wget_wstr(WINDOW *win, wint_t *wstr);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The <code>get_wstr()</code> and <code>wget_wstr()</code> functions get a wide character string from the terminal associated with the window <code>stdscr</code> or window <code>win</code>, respectively. The <code>mvget_wstr()</code> and <code>mvwget_wstr()</code> functions move the cursor to the position specified in <code>stdscr</code> or <code>win</code>, respectively, then get a wide character string.</p> <p>These functions call <code>wget_wch(3XCURSES)</code> and place each received character in <code>wstr</code> until a newline character, end-of-line character, or end-of-file character is received, which is also placed in <code>wstr</code>. The erase and kill characters set by the user are processed.</p> <p>The <code>getn_wstr()</code>, <code>mvgetn_wstr()</code>, <code>mvwgetn_wstr()</code> and <code>wgetn_wstr()</code> functions read at most <code>n</code> characters. These functions are used to prevent overflowing the input buffer.</p> | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>wstr</i></td><td>Is a pointer to the area where the character string is to be placed.</td></tr><tr><td><i>n</i></td><td>Is the maximum number of characters to read from input.</td></tr><tr><td><i>y</i></td><td>Is the y (row) coordinate of starting position of character string to be read.</td></tr><tr><td><i>x</i></td><td>Is the x (column) coordinate of starting position of character string to be read.</td></tr><tr><td><i>win</i></td><td>points to the window associated with the terminal from which the character is to be read.</td></tr></table> | <i>wstr</i> | Is a pointer to the area where the character string is to be placed. | <i>n</i> | Is the maximum number of characters to read from input. | <i>y</i> | Is the y (row) coordinate of starting position of character string to be read. | <i>x</i> | Is the x (column) coordinate of starting position of character string to be read. | <i>win</i> | points to the window associated with the terminal from which the character is to be read. |
| <i>wstr</i> | Is a pointer to the area where the character string is to be placed. | | | | | | | | | | |
| <i>n</i> | Is the maximum number of characters to read from input. | | | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of starting position of character string to be read. | | | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of starting position of character string to be read. | | | | | | | | | | |
| <i>win</i> | points to the window associated with the terminal from which the character is to be read. | | | | | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | | | | | |
| ERRORS | None. | | | | | | | | | | |

getn_wstr(3XCURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `get_wch(3XCURSES)`, `getnstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

getparyx(3CURSES)

| NAME | <code> curs_getyx, getyx, getparyx, getbegyx, getmaxyx</code> – get curses cursor and window coordinates | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> void getyx(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>); void getparyx(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>); void getbegyx(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>); void getmaxyx(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>);</pre> | | | | |
| DESCRIPTION | <p>With the <code>getyx()</code> macro, the cursor position of the window is placed in the two integer variables <i>y</i> and <i>x</i>.</p> <p>With the <code>getparyx()</code> macro, if <i>win</i> is a subwindow, the beginning coordinates of the subwindow relative to the parent window are placed into two integer variables, <i>y</i> and <i>x</i>. Otherwise, <code>-1</code> is placed into <i>y</i> and <i>x</i>.</p> <p>Like <code>getyx()</code>, the <code>getbegyx()</code> and <code>getmaxyx()</code> macros store the current beginning coordinates and size of the specified window.</p> | | | | |
| RETURN VALUES | The return values of these macros are undefined (that is, they should not be used as the right-hand side of assignment statements). | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that all of these interfaces are macros and that “&” is not necessary before the variables <i>y</i> and <i>x</i>.</p> | | | | |

| NAME | getbegyx, getmaxyx, getparyx, getyx – get cursor or window coordinates | | | | | | |
|----------------------|--|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> void getbegyx(WINDOW *win, int y, int x); void getmaxyx(WINDOW *win, int y, int x); void getparyx(WINDOW *win, int y, int x); void getyx(WINDOW *win, int y, int x);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>getyx()</code> macro stores the current cursor position of the specified window in <code>x</code> and <code>y</code>.</p> <p>The <code>getparyx()</code> macro stores the <code>x</code> and <code>y</code> coordinates (relative to the parent window) of the specified window's origin (upper-left corner). If <code>win</code> does not point to a subwindow, <code>x</code> and <code>y</code> are set to <code>-1</code>.</p> <p>The <code>getbegyx()</code> macro stores the <code>x</code> and <code>y</code> coordinates of the specified window's origin (upper-left corner).</p> <p>The <code>getmaxyx()</code> macro stores the numbers of rows in the specified window in <code>y</code> and the number of columns in <code>x</code>.</p> | | | | | | |
| PARAMETERS | <p><code>win</code> Is a pointer to a window.</p> <p><code>y</code> stores the <code>y</code> coordinate for the cursor or origin. The <code>getmaxyx()</code> macro uses it to store the number of rows in the window.</p> <p><code>x</code> stores the <code>x</code> coordinate for the cursor or origin. The <code>getmaxyx()</code> macro uses it to store the number of columns in the window.</p> | | | | | | |
| RETURN VALUES | These macros do not return a value. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

getstr(3CURSES)

| NAME | <code> curs_getstr, getstr, wgetstr, mvgetstr, mvwgetstr, wgetnstr </code> – get character strings from curses terminal keyboard | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int getstr(char *<i>str</i>); int wgetstr(WINDOW *<i>win</i>, char *<i>str</i>); int mvgetstr(int <i>y</i>, int <i>x</i>, char *<i>str</i>); int mvwgetstr(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>, char *<i>str</i>); int wgetnstr(WINDOW *<i>win</i>, char *<i>str</i>, int <i>n</i>);</pre> | | | | |
| DESCRIPTION | The effect of <code>getstr()</code> is as though a series of calls to <code>getch()</code> were made, until a newline or carriage return is received. The resulting value is placed in the area pointed to by the character pointer <i>str</i> . <code>wgetnstr()</code> reads at most <i>n</i> characters, thus preventing a possible overflow of the input buffer. The user's erase and kill characters are interpreted, as well as any special keys (such as function keys, HOME key, and CLEAR key.) | | | | |
| RETURN VALUES | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code> curs_getch(3CURSES), curses(3CURSES), attributes(5) </code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . Note that <code>getstr()</code> , <code>mvgetstr()</code> , and <code>mvwgetstr()</code> may be macros. | | | | |

| | | | | | | | | | | | |
|--------------------|--|------------|--|----------|---|----------|--|----------|---|------------|---|
| NAME | getnstr, getstr, mvgetnstr, mvgetstr, mvwgetnstr, mvwgetstr, wgetnstr, wgetstr – get a multibyte character string from terminal | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int getnstr(char *str, int n); int getstr(char *str); int mvgetnstr(int y, int x, char *str, int n); int mvgetstr(int y, int x, char *str); int mvwgetnstr(WINDOW *win, int y, int x, char *str, int n); int mvwgetstr(WINDOW *win, int y, int x, char *str); int wgetnstr(WINDOW *win, char *str, int n); int wgetstr(WINDOW *win, char *str);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The <code>getstr()</code> and <code>wgetstr()</code> functions get a character string from the terminal associated with the window <code>stdscr</code> or window <code>win</code>, respectively. The <code>mvgetstr()</code> and <code>mvwgetstr()</code> functions move the cursor to the position specified in <code>stdscr</code> or <code>win</code>, respectively, then get a character string.</p> <p>These functions call <code>wgetch(3XCURSES)</code> and place each received character in <code>str</code> until a newline is received, which is also placed in <code>str</code>. The erase and kill characters set by the user are processed.</p> <p>The <code>getnstr()</code>, <code>mvgetnstr()</code>, <code>mvwgetnstr()</code> and <code>wgetnstr()</code> functions read at most <code>n</code> characters. These functions are used to prevent overflowing the input buffer.</p> <p>The <code>getnstr()</code>, <code>wgetnstr()</code>, <code>mvgetnstr()</code>, and <code>mvwgetnstr()</code> functions only return complete multibyte characters. If the area pointed to by <code>str</code> is not large enough to hold at least one character, these functions fail.</p> | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>str</i></td> <td>Is a pointer to the area where the character string is to be placed.</td> </tr> <tr> <td><i>n</i></td> <td>Is the maximum number of characters to read from input.</td> </tr> <tr> <td><i>y</i></td> <td>Is the y (row) coordinate of starting position of character string to be read.</td> </tr> <tr> <td><i>x</i></td> <td>Is the x (column) coordinate of starting position of character string to be read.</td> </tr> <tr> <td><i>win</i></td> <td>Points to the window associated with the terminal from which the character is to be read.</td> </tr> </table> | <i>str</i> | Is a pointer to the area where the character string is to be placed. | <i>n</i> | Is the maximum number of characters to read from input. | <i>y</i> | Is the y (row) coordinate of starting position of character string to be read. | <i>x</i> | Is the x (column) coordinate of starting position of character string to be read. | <i>win</i> | Points to the window associated with the terminal from which the character is to be read. |
| <i>str</i> | Is a pointer to the area where the character string is to be placed. | | | | | | | | | | |
| <i>n</i> | Is the maximum number of characters to read from input. | | | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of starting position of character string to be read. | | | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of starting position of character string to be read. | | | | | | | | | | |
| <i>win</i> | Points to the window associated with the terminal from which the character is to be read. | | | | | | | | | | |

getstr(3XCURSES)

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `getch(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | curs_kernel, def_prog_mode, def_shell_mode, reset_prog_mode, reset_shell_mode, resetty, savetty, getsyx, setsyx, ripoffline, curs_set, napms – low-level curses routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int def_prog_mode(void); int def_shell_mode(void); int reset_prog_mode(void); int reset_shell_mode(void); int resetty(void); int savetty(void); int getsyx(int y, int x); int setsyx(int y, int x); int ripoffline(int line, int (*init)(WINDOW *, int)); int curs_set(int visibility); int napms(int ms);</pre> |
| DESCRIPTION | <p>The following routines give low-level access to various curses functionality. These routines typically are used inside library routines.</p> <p>The <code>def_prog_mode()</code> and <code>def_shell_mode()</code> routines save the current terminal modes as the “program” (in curses) or “shell” (not in curses) state for use by the <code>reset_prog_mode()</code> and <code>reset_shell_mode()</code> routines. This is done automatically by <code>initscr()</code>.</p> <p>The <code>reset_prog_mode()</code> and <code>reset_shell_mode()</code> routines restore the terminal to “program” (in curses) or “shell” (out of curses) state. These are done automatically by <code>endwin()</code> and, after an <code>endwin()</code>, by <code>doupdate()</code>, so they normally are not called.</p> <p>The <code>resetty()</code> and <code>savetty()</code> routines save and restore the state of the terminal modes. <code>savetty()</code> saves the current state in a buffer and <code>resetty()</code> restores the state to what it was at the last call to <code>savetty()</code>.</p> <p>With the <code>getsyx()</code> routine, the current coordinates of the virtual screen cursor are returned in <code>y</code> and <code>x</code>. If <code>leaveok()</code> is currently TRUE, then <code>-1,-1</code> is returned. If lines have been removed from the top of the screen, using <code>ripoffline()</code>, <code>y</code> and <code>x</code> include these lines; therefore, <code>y</code> and <code>x</code> should be used only as arguments for <code>setsyx()</code>.</p> <p>With the <code>setsyx()</code> routine, the virtual screen cursor is set to <code>y, x</code>. If <code>y</code> and <code>x</code> are both <code>-1</code>, then <code>leaveok()</code> is set. The two routines <code>getsyx()</code> and <code>setsyx()</code> are designed to be used by a library routine, which manipulates curses windows but does not</p> |

getsyx(3CURSES)

want to change the current position of the program's cursor. The library routine would call `getsyx()` at the beginning, do its manipulation of its own windows, do a `wnoutrefresh()` on its windows, call `setsyx()`, and then call `doupdate()`.

The `riporffline()` routine provides access to the same facility that `slk_init()` (see `curs_slk(3CURSES)`) uses to reduce the size of the screen. `riporffline()` must be called before `initscr()` or `newterm()` is called. If `line` is positive, a line is removed from the top of `stdscr()`; if `line` is negative, a line is removed from the bottom. When this is done inside `initscr()`, the routine `init()` (supplied by the user) is called with two arguments: a window pointer to the one-line window that has been allocated and an integer with the number of columns in the window. Inside this initialization routine, the integer variables `LINES` and `COLS` (defined in `< curses.h >`) are not guaranteed to be accurate and `wrefresh()` or `doupdate()` must not be called. It is allowable to call `wnoutrefresh()` during the initialization routine.

`riporffline()` can be called up to five times before calling `initscr()` or `newterm()`.

With the `curs_set()` routine, the cursor state is set to invisible, normal, or very visible for *visibility* equal to 0, 1, or 2 respectively. If the terminal supports the *visibility* requested, the previous *cursor* state is returned; otherwise, `ERR` is returned.

The `napms()` routine is used to sleep for *ms* milliseconds.

RETURN VALUES

Except for `curs_set()`, these routines always return `OK`. `curs_set()` returns the previous cursor state, or `ERR` if the requested *visibility* is not supported.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_initscr(3CURSES)`, `curs_outopts(3CURSES)`, `curs_refresh(3CURSES)`, `curs_scr_dump(3CURSES)`, `curs_slk(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES

The header `< curses.h >` automatically includes the headers `< stdio.h >` and `< unctrl.h >`.

Note that `getsyx()` is a macro, so an ampersand (&) is not necessary before the variables *y* and *x*.

| NAME | curs_getwch, getwch, wgetwch, mvgetwch, mvwgetwch, ungetwch – get (or push back) wchar_t characters from curses terminal keyboard | | | | |
|----------------------|--|-------------|-----------------|-----------|-----------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int getwch(void); int wgetwch(WINDOW *win); int mvgetwch(int y, int x); int mvwgetwch(WINDOW *win, int y, int x); int ungetwch(int wch);</pre> | | | | |
| DESCRIPTION | <p>The <code>getwch()</code>, <code>wgetwch()</code>, <code>mvgetwch()</code>, and <code>mvwgetwch()</code> routines read an EUC character from the terminal associated with the window, transform it into a <code>wchar_t</code> character, and return a <code>wchar_t</code> character. In no-delay mode, if no input is waiting, the value <code>ERR</code> is returned. In delay mode, the program waits until the system passes text through to the program. Depending on the setting of <code>cbreak</code>, this is after one character (<code>cbreak</code> mode), or after the first newline (<code>nocbreak</code> mode). In half-delay mode, the program waits until a character is typed or the specified timeout has been reached. Unless <code>noecho</code> has been set, the character will also be echoed into the designated window.</p> <p>If the window is not a pad, and it has been moved or modified since the last call to <code>wrefresh(3CURSES)</code>, <code>wrefresh</code> will be called before another character is read.</p> <p>If <code>keypad</code> is <code>TRUE</code>, and a function key is pressed, the token for that function key is returned instead of the raw characters. Possible function keys are defined in <code><curses.h></code> with integers beginning with <code>0401</code>, whose names begin with <code>KEY_</code>. If a character that could be the beginning of a function key (such as escape) is received, <code>curses(3CURSES)</code> sets a timer. If the remainder of the sequence does not come in within the designated time, the character is passed through; otherwise, the function key value is returned. For this reason, many terminals experience a delay between the time a user presses the escape key and the escape is returned to the program.</p> <p>The <code>ungetwch()</code> routine places <code>wch</code> back onto the input queue to be returned by the next call to <code>wgetwch()</code>.</p> | | | | |
| Function Keys | <p>The following function keys, defined in <code><curses.h></code>, might be returned by <code>getwch()</code> if <code>keypad</code> has been enabled. Note that not all of these may be supported on a particular terminal if the terminal does not transmit a unique code when the key is pressed or if the definition for the key is not present in the <code>terminfo(4)</code> database.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;"><i>Name</i></th> <th style="text-align: left; padding: 2px;"><i>Key name</i></th> </tr> </thead> <tbody> <tr> <td style="padding: 2px;">KEY_BREAK</td> <td style="padding: 2px;">Break key</td> </tr> </tbody> </table> | <i>Name</i> | <i>Key name</i> | KEY_BREAK | Break key |
| <i>Name</i> | <i>Key name</i> | | | | |
| KEY_BREAK | Break key | | | | |

getwch(3CURSES)

| <i>Name</i> | <i>Key name</i> |
|-------------------|---|
| KEY_DOWN | The four arrow keys . . . |
| KEY_UP | |
| KEY_LEFT | |
| KEY_RIGHT | |
| KEY_HOME | Home key (upward+left arrow) |
| KEY_BACKSPACE | Backspace |
| KEY_F0 | Function keys; space for 64 keys is reserved. |
| KEY_F(<i>n</i>) | For $0 \leq n \leq 63$ |
| KEY_DL | Delete line |
| KEY_IL | Insert line |
| KEY_DC | Delete character |
| KEY_IC | Insert char or enter insert mode |
| KEY_EIC | Exit insert char mode |
| KEY_CLEAR | Clear screen |
| KEY_EOS | Clear to end of screen |
| KEY_EOL | Clear to end of line |
| KEY_SF | Scroll 1 line forward |
| KEY_SR | Scroll 1 line backward (reverse) |
| KEY_NPAGE | Next page |
| KEY_PPAGE | Previous page |
| KEY_STAB | Set tab |
| KEY_CTAB | Clear tab |
| KEY_CATAB | Clear all tabs |
| KEY_ENTER | Enter or send |
| KEY_SRESET | Soft (partial) reset |
| KEY_RESET | Reset or hard reset |
| KEY_PRINT | Print or copy |
| KEY_LL | Home down or bottom (lower left). Keypad is arranged like this: A1 up A3 left B2 right C1 down C3 |

| <i>Name</i> | <i>Key name</i> |
|---------------|-----------------------|
| KEY_A1 | Upper left of keypad |
| KEY_A3 | Upper right of keypad |
| KEY_B2 | Center of keypad |
| KEY_C1 | Lower left of keypad |
| KEY_C3 | Lower right of keypad |
| KEY_BTAB | Back tab key |
| KEY_BEG | Beg(inning) key |
| KEY_CANCEL | Cancel key |
| KEY_CLOSE | Close key |
| KEY_COMMAND | Cmd (command) key |
| KEY_COPY | Copy key |
| KEY_CREATE | Create key |
| KEY_END | End key |
| KEY_EXIT | Exit key |
| KEY_FIND | Find key |
| KEY_HELP | Help key |
| KEY_MARK | Mark key |
| KEY_MESSAGE | Message key |
| KEY_MOVE | Move key |
| KEY_NEXT | Next object key |
| KEY_OPEN | Open key |
| KEY_OPTIONS | Options key |
| KEY_PREVIOUS | Previous object key |
| KEY_REDO | Redo key |
| KEY_REFERENCE | Reference key |
| KEY_REFRESH | Refresh key |
| KEY_REPLACE | Replace key |
| KEY_RESTART | Restart key |
| KEY_RESUME | Resume key |

getwch(3CURSES)

| <i>Name</i> | <i>Key name</i> |
|---------------|-------------------------|
| KEY_SAVE | Save key |
| KEY_SBEG | Shifted beginning key |
| KEY_SCANCEL | Shifted cancel key |
| KEY_SCOMMAND | Shifted command key |
| KEY_SCOPY | Shifted copy key |
| KEY_SCREATE | Shifted create key |
| KEY_SDC | Shifted delete char key |
| KEY_SDL | Shifted delete line key |
| KEY_SELECT | Select key |
| KEY_SEND | Shifted end key |
| KEY_SEOL | Shifted clear line key |
| KEY_SEXIT | Shifted exit key |
| KEY_SFIND | Shifted find key |
| KEY_SHELP | Shifted help key |
| KEY_SHOME | Shifted home key |
| KEY_SIC | Shifted input key |
| KEY_SLEFT | Shifted left arrow key |
| KEY_SMESSAGE | Shifted message key |
| KEY_SMOVE | Shifted move key |
| KEY_SNEXT | Shifted next key |
| KEY_SOPTIONS | Shifted options key |
| KEY_SPREVIOUS | Shifted prev key |
| KEY_SPRINT | Shifted print key |
| KEY_SREDO | Shifted redo key |
| KEY_SREPLACE | Shifted replace key |
| KEY_SRIGHT | Shifted right arrow |
| KEY_SRSUME | Shifted resume key |
| KEY_SSAVE | Shifted save key |
| KEY_SSUSPEND | Shifted suspend key |

| <i>Name</i> | <i>Key name</i> |
|-------------|------------------|
| KEY_SUNDO | Shifted undo key |
| KEY_SUSPEND | Suspend key |
| KEY_UNDO | Undo key |

RETURN VALUE All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), curs_inopts(3CURSES), curs_move(3CURSES), wrefresh(3CURSES), terminfo(4), attributes(5)

NOTES The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <widec.h>.

Use of the escape key by a programmer for a single character function is discouraged.

When using getwch(), wgetwch(), mvgetwch(), or mvwgetwch(), nocbreak mode and echo mode should not be used at the same time. Depending on the state of the tty driver when each character is typed, the program may produce undesirable results.

Note that getwch(), mvgetwch(), and mvwgetwch() may be macros.

get_wch(3XCURSES)

| | |
|--------------------|---|
| NAME | get_wch, wget_wch, mvget_wch, mvwget_wch – get a wide character from terminal |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int get_wch(wint_t *ch); int wget_wch(WINDOW *win, wint_t *ch); int mvget_wch(int y, int x, wint_t *ch); int mvwget_wch(WINDOW *win, int y, int x, wint_t *ch);</pre> |
| DESCRIPTION | <p>The <code>get_wch()</code> and <code>wget_wch()</code> functions get a wide character from the terminal associated with the window <code>stdscr</code> or window <code>win</code>, respectively. The <code>mvget_wch()</code> and <code>mvwget_wch()</code> functions move the cursor to the position specified in <code>stdscr</code> or <code>win</code>, respectively, then get a character.</p> <p>If the window is not a pad and has been changed since the last call to <code>refresh(3XCURSES)</code>, <code>get_wch()</code> calls <code>refresh()</code> to update the window before the next character is read.</p> <p>The setting of certain functions affects the behavior of the <code>get_wch()</code> set of functions. For example, if <code>cbreak(3XCURSES)</code> is set, characters typed by the user are immediately processed. If <code>halfdelay(3XCURSES)</code> is set, <code>get_wch()</code> waits until a character is typed or returns <code>ERR</code> if no character is typed within the specified timeout period. This timeout can also be specified for individual windows with the <i>delay</i> parameter of <code>timeout(3XCURSES)</code>. A negative value waits for input; a value of 0 returns <code>ERR</code> if no input is ready; a positive value blocks until input arrives or the time specified expires (in which case <code>ERR</code> is returned). If <code>nodelay(3XCURSES)</code> is set, <code>ERR</code> is returned if no input is waiting; if not set, <code>get_wch()</code> waits until input arrives. Each character will be echoed to the window unless <code>noecho(3XCURSES)</code> has been set.</p> <p>If keypad handling is enabled (<code>keypad(3XCURSES)</code> is <code>TRUE</code>), the token for the function key (a <code>KEY_</code> value) is stored in the object pointed to by <code>ch</code> and <code>KEY_CODE_YES</code> is returned. If a character is received that could be the beginning of a function key (for example, <code>ESC</code>), an inter-byte timer is set. If the remainder of the sequence is not received before the time expires, the character is passed through; otherwise, the value of the function key is returned. If <code>notimeout()</code> is set, the inter-byte timer is not used.</p> <p>The <code>ESC</code> key is typically a prefix key used with function keys and should not be used as a single character.</p> <p>See the <code>getch(3XCURSES)</code> manual page for a list of tokens for function keys that are returned by the <code>get_wch()</code> set of functions if keypad handling is enabled (Some terminals may not support all tokens).</p> |

get_wch(3XCURSES)

PARAMETERS

ch Is a pointer to a wide integer where the returned wide character or KEY_ value can be stored.

win Is a pointer to the window associated with the terminal from which the character is to be read.

y Is the y (row) coordinate for the position of the character to be read.

x Is the x (column) coordinate for the position of the character to be read.

RETURN VALUES When these functions successfully report the pressing of a function key, they return KEY_CODE_YES. When they successfully report a wide character, they return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO cbreak(3XCURSES), echo(3XCURSES), halfdelay(3XCURSES), keypad(3XCURSES), libcurses(3XCURSES), nodelay(3XCURSES), notimeout(3XCURSES), raw(3XCURSES), timeout(3XCURSES), attributes(5), standards(5)

getwin(3CURSES)

| | |
|--------------------|---|
| NAME | curs_util, unctrl, keyname, filter, use_env, putwin, getwin, delay_output, flushingp – curses miscellaneous utility routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> char *unctrl (chtype <i>c</i>); char *keyname (int <i>c</i>); int filter (void); void use_env (char <i>bool</i>); int putwin (WINDOW *<i>win</i>, FILE *<i>filep</i>); WINDOW *getwin (FILE *<i>filep</i>); int delay_output (int <i>ms</i>); int flushingp (void);</pre> |
| DESCRIPTION | <p>The unctrl () macro expands to a character string which is a printable representation of the character <i>c</i>. Control characters are displayed in the ^X notation. Printing characters are displayed as is.</p> <p>With the keyname () routine, a character string corresponding to the key <i>c</i> is returned.</p> <p>The filter () routine, if used, is called before initscr () or newterm () are called. It makes curses think that there is a one-line screen. curses does not use any terminal capabilities that assume that they know on what line of the screen the cursor is positioned.</p> <p>The use_env () routine, if used, is called before initscr () or newterm () are called. When called with FALSE as an argument, the values of lines and columns specified in the terminfo database will be used, even if environment variables LINES and COLUMNS (used by default) are set, or if curses is running in a window (in which case default behavior would be to use the window size if LINES and COLUMNS are not set).</p> <p>With the putwin () routine, all data associated with window <i>win</i> is written into the file to which <i>filep</i> points. This information can be later retrieved using the getwin () function.</p> <p>The getwin () routine reads window related data stored in the file by putwin (). The routine then creates and initializes a new window using that data. It returns a pointer to the new window.</p> <p>The delay_output () routine inserts an <i>ms</i> millisecond pause in output. This routine should not be used extensively because padding characters are used rather than a CPU pause.</p> <p>The flushingp () routine throws away any typeahead that has been typed by the user and has not yet been read by the program.</p> |

getwin(3CURSES)

RETURN VALUES Except for `flushinp()`, routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`flushinp()` always returns `OK`.

Routines that return pointers return `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_initscr(3CURSES)`, `curl_scr_dump(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `unctrl()` is a macro, which is defined in `<unctrl.h>`.

getwin(3XCURSES)

NAME getwin, putwin – read a window from, and write a window to, a file

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

WINDOW *getwin(FILE *filep);

int putwin(WINDOW *win, FILE *filep);
```

DESCRIPTION

The `getwin()` function reads window-related data (written earlier by `putwin()`) from the `stdio` stream pointed to by `filep`. It then creates and initializes a new window using that data.

The `putwin()` function writes all the data associated with the window pointed to by `win` to the `stdio` stream pointed to by `filep`. The `getwin()` function can later retrieve this data.

PARAMETERS

`filep` Is a pointer to a `stdio` stream.

`win` Is a pointer to a window.

RETURN VALUES

On success, the `getwin()` function returns a pointer to the new window created. Otherwise, it returns a null pointer.

On success, the `putwin()` function returns `OK`. Otherwise, it returns `ERR`.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `libcurses(3XCURSES)`, `scr_dump(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_getwstr, getwstr, getnwstr, wgetwstr, wgetnwstr, mvgetwstr, mvgetnwstr, mvwgetwstr, mvwgetnwstr – get wchar_t character strings from curses terminal keyboard | | | | |
|---------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int getwstr(wchar_t *wstr); int getnwstr(wchar_t *wstr, int n); int wgetwstr(WINDOW *win, wchar_t *wstr); int wgetnwstr(WINDOW *win, wchar_t *wstr, int n); int mvgetwstr(int y, int x, wchar_t *wstr); int mvgetnwstr(int y, int x, wchar_t *wstr, int n); int mvwgetwstr(WINDOW *win, int y, int x, wchar_t *wstr); int mvwgetnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);</pre> | | | | |
| DESCRIPTION | The effect of getwstr() is as though a series of calls to getch(3CURSES) were made, until a newline and carriage return is received. The resulting value is placed in the area pointed to by the wchar_t pointer wstr. getnwstr() reads at most n wchar_t characters, thus preventing a possible overflow of the input buffer. The user's erase and kill characters are interpreted, as well as any special keys (such as function keys, HOME key, CLEAR key, etc.). | | | | |
| RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See attributes(5) for a description of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), getch(3CURSES), attributes(5) | | | | |
| NOTES | <p>The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h>, and <wchar.h>.</p> <p>Note that all routines except wgetnwstr() may be macros.</p> | | | | |

get_wstr(3XCURSES)

| | | | | | | | | | | | |
|----------------------|---|-------------|--|----------|---|----------|--|----------|---|------------|---|
| NAME | getn_wstr, get_wstr, mvgetn_wstr, mvget_wstr, mvwgetn_wstr, mvwget_wstr, wgetn_wstr, wget_wstr – get a wide character string from terminal | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int getn_wstr(wint_t *wstr, int n); int get_wstr(wint_t *wstr); int mvgetn_wstr(int y, int x, wint_t *wstr, int n); int mvget_wstr(int y, int x, wint_t *wstr); int mvwgetn_wstr(WINDOW *win, int y, int x, wint_t *wstr, int n); int mvwget_wstr(WINDOW *win, int y, int x, wint_t *wstr); int wgetn_wstr(WINDOW *win, wint_t *wstr, int n); int wget_wstr(WINDOW *win, wint_t *wstr);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The <code>get_wstr()</code> and <code>wget_wstr()</code> functions get a wide character string from the terminal associated with the window <code>stdscr</code> or window <code>win</code>, respectively. The <code>mvget_wstr()</code> and <code>mvwget_wstr()</code> functions move the cursor to the position specified in <code>stdscr</code> or <code>win</code>, respectively, then get a wide character string.</p> <p>These functions call <code>wget_wch(3XCURSES)</code> and place each received character in <code>wstr</code> until a newline character, end-of-line character, or end-of-file character is received, which is also placed in <code>wstr</code>. The erase and kill characters set by the user are processed.</p> <p>The <code>getn_wstr()</code>, <code>mvgetn_wstr()</code>, <code>mvwgetn_wstr()</code> and <code>wgetn_wstr()</code> functions read at most <code>n</code> characters. These functions are used to prevent overflowing the input buffer.</p> | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>wstr</i></td><td>Is a pointer to the area where the character string is to be placed.</td></tr><tr><td><i>n</i></td><td>Is the maximum number of characters to read from input.</td></tr><tr><td><i>y</i></td><td>Is the y (row) coordinate of starting position of character string to be read.</td></tr><tr><td><i>x</i></td><td>Is the x (column) coordinate of starting position of character string to be read.</td></tr><tr><td><i>win</i></td><td>points to the window associated with the terminal from which the character is to be read.</td></tr></table> | <i>wstr</i> | Is a pointer to the area where the character string is to be placed. | <i>n</i> | Is the maximum number of characters to read from input. | <i>y</i> | Is the y (row) coordinate of starting position of character string to be read. | <i>x</i> | Is the x (column) coordinate of starting position of character string to be read. | <i>win</i> | points to the window associated with the terminal from which the character is to be read. |
| <i>wstr</i> | Is a pointer to the area where the character string is to be placed. | | | | | | | | | | |
| <i>n</i> | Is the maximum number of characters to read from input. | | | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of starting position of character string to be read. | | | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of starting position of character string to be read. | | | | | | | | | | |
| <i>win</i> | points to the window associated with the terminal from which the character is to be read. | | | | | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | | | | | |
| ERRORS | None. | | | | | | | | | | |

get_wstr(3XCURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `get_wch(3XCURSES)`, `getnstr(3XCURSES)`, `libcurses(3XCURSES)`,
`attributes(5)`, `standards(5)`

getyx(3CURSES)

| NAME | <code> curs_getyx, getyx, getparyx, getbegyx, getmaxyx</code> – get curses cursor and window coordinates | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> void getyx(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>); void getparyx(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>); void getbegyx(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>); void getmaxyx(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>);</pre> | | | | |
| DESCRIPTION | <p>With the <code>getyx()</code> macro, the cursor position of the window is placed in the two integer variables <i>y</i> and <i>x</i>.</p> <p>With the <code>getparyx()</code> macro, if <i>win</i> is a subwindow, the beginning coordinates of the subwindow relative to the parent window are placed into two integer variables, <i>y</i> and <i>x</i>. Otherwise, <code>-1</code> is placed into <i>y</i> and <i>x</i>.</p> <p>Like <code>getyx()</code>, the <code>getbegyx()</code> and <code>getmaxyx()</code> macros store the current beginning coordinates and size of the specified window.</p> | | | | |
| RETURN VALUES | The return values of these macros are undefined (that is, they should not be used as the right-hand side of assignment statements). | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that all of these interfaces are macros and that “&” is not necessary before the variables <i>y</i> and <i>x</i>.</p> | | | | |

| NAME | getbegyx, getmaxyx, getparyx, getyx – get cursor or window coordinates | | | | | | |
|----------------------|--|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> void getbegyx(WINDOW *win, int y, int x); void getmaxyx(WINDOW *win, int y, int x); void getparyx(WINDOW *win, int y, int x); void getyx(WINDOW *win, int y, int x);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>getyx()</code> macro stores the current cursor position of the specified window in <code>x</code> and <code>y</code>.</p> <p>The <code>getparyx()</code> macro stores the <code>x</code> and <code>y</code> coordinates (relative to the parent window) of the specified window's origin (upper-left corner). If <code>win</code> does not point to a subwindow, <code>x</code> and <code>y</code> are set to <code>-1</code>.</p> <p>The <code>getbegyx()</code> macro stores the <code>x</code> and <code>y</code> coordinates of the specified window's origin (upper-left corner).</p> <p>The <code>getmaxyx()</code> macro stores the numbers of rows in the specified window in <code>y</code> and the number of columns in <code>x</code>.</p> | | | | | | |
| PARAMETERS | <p><code>win</code> Is a pointer to a window.</p> <p><code>y</code> stores the <code>y</code> coordinate for the cursor or origin. The <code>getmaxyx()</code> macro uses it to store the number of rows in the window.</p> <p><code>x</code> stores the <code>x</code> coordinate for the cursor or origin. The <code>getmaxyx()</code> macro uses it to store the number of columns in the window.</p> | | | | | | |
| RETURN VALUES | These macros do not return a value. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

halfdelay(3CURSES)

| | |
|--------------------|---|
| NAME | <code>curs_inopts</code> , <code>cbreak</code> , <code>nocbreak</code> , <code>echo</code> , <code>noecho</code> , <code>halfdelay</code> , <code>intrflush</code> , <code>keypad</code> , <code>meta</code> , <code>nodelay</code> , <code>notimeout</code> , <code>raw</code> , <code>noraw</code> , <code>noqiflush</code> , <code>qiflush</code> , <code>timeout</code> , <code>wtimeout</code> , <code>typeahead</code> – curses terminal input option control routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int cbreak(void); int nocbreak(void); int echo(void); int noecho(void); int halfdelay(int <i>tenths</i>); int intrflush(WINDOW *<i>win</i>, bool <i>bf</i>); int keypad(WINDOW *<i>win</i>, bool <i>bf</i>); int meta(WINDOW *<i>win</i>, bool <i>bf</i>); int nodelay(WINDOW *<i>win</i>, bool <i>bf</i>); int notimeout(WINDOW *<i>win</i>, bool <i>bf</i>); int raw(void); int noraw(void); void noqiflush(void); void qiflush(void); void timeout(int <i>delay</i>); void wtimeout(WINDOW *<i>win</i>, int <i>delay</i>); int typeahead(int <i>fildes</i>);</pre> |
| DESCRIPTION | <p>The <code>cbreak()</code> and <code>nocbreak()</code> routines put the terminal into and out of <code>cbreak()</code> mode, respectively. In this mode, characters typed by the user are immediately available to the program, and erase/kill character-processing is not performed. When out of this mode, the tty driver buffers the typed characters until a newline or carriage return is typed. Interrupt and flow control characters are unaffected by this mode. Initially the terminal may or may not be in <code>cbreak()</code> mode, as the mode is inherited; therefore, a program should call <code>cbreak()</code> or <code>nocbreak()</code> explicitly. Most interactive programs using curses set the <code>cbreak()</code> mode.</p> <p>Note that <code>cbreak()</code> overrides <code>raw()</code>. (See <code>curs_getch(3CURSES)</code> for a discussion of how these routines interact with <code>echo()</code> and <code>noecho()</code>.)</p> <p>The <code>echo()</code> and <code>noecho()</code> routines control whether characters typed by the user are echoed by <code>getch()</code> as they are typed. Echoing by the tty driver is always disabled, but initially <code>getch()</code> is in echo mode, so characters typed are echoed. Authors of</p> |

most interactive programs prefer to do their own echoing in a controlled area of the screen, or not to echo at all, so they disable echoing by calling `noecho()`. (See `curs_getch(3CURSES)` for a discussion of how these routines interact with `cbreak()` and `nocbreak()`.)

The `halfdelay()` routine is used for half-delay mode, which is similar to `cbreak()` mode in that characters typed by the user are immediately available to the program. However, after blocking for *tenths* tenths of seconds, `ERR` is returned if nothing has been typed. The value of *tenths* must be a number between 1 and 255. Use `nocbreak()` to leave half-delay mode.

If the `intrflush()` option is enabled, (*bf* is `TRUE`), when an interrupt key is pressed on the keyboard (interrupt, break, quit) all output in the tty driver queue will be flushed, giving the effect of faster response to the interrupt, but causing curses to have the wrong idea of what is on the screen. Disabling (*bf* is `FALSE`), the option prevents the flush. The default for the option is inherited from the tty driver settings. The window argument is ignored.

The `keypad()` option enables the keypad of the user's terminal. If enabled (*bf* is `TRUE`), the user can press a function key (such as an arrow key) and `wgetch()` returns a single value representing the function key, as in `KEY_LEFT`. If disabled (*bf* is `FALSE`), curses does not treat function keys specially and the program has to interpret the escape sequences itself. If the keypad in the terminal can be turned on (made to transmit) and off (made to work locally), turning on this option causes the terminal keypad to be turned on when `wgetch()` is called. The default value for keypad is false.

Initially, whether the terminal returns 7 or 8 significant bits on input depends on the control mode of the tty driver (see `termio(7I)`). To force 8 bits to be returned, invoke `meta(win, TRUE)`. To force 7 bits to be returned, invoke `meta(win, FALSE)`. The window argument, *win*, is always ignored. If the terminfo capabilities `smm` (`meta_on`) and `rmm` (`meta_off`) are defined for the terminal, `smm` is sent to the terminal when `meta(win, TRUE)` is called and `rmm` is sent when `meta(win, FALSE)` is called.

The `nodelay()` option causes `getch()` to be a non-blocking call. If no input is ready, `getch()` returns `ERR`. If disabled (*bf* is `FALSE`), `getch()` waits until a key is pressed.

While interpreting an input escape sequence, `wgetch()` sets a timer while waiting for the next character. If `notimeout(win, TRUE)` is called, then `wgetch()` does not set a timer. The purpose of the timeout is to differentiate between sequences received from a function key and those typed by a user.

With the `raw()` and `noraw()` routines, the terminal is placed into or out of raw mode. Raw mode is similar to `cbreak()` mode, in that characters typed are immediately passed through to the user program. The differences are that in raw mode, the interrupt, quit, suspend, and flow control characters are all passed through uninterpreted, instead of generating a signal. The behavior of the `BREAK` key depends on other bits in the tty driver that are not set by `curses`.

halfdelay(3CURSES)

When the `noqiflush()` routine is used, normal flush of input and output queues associated with the `INTR`, `QUIT` and `SUSP` characters will not be done (see `termio(7I)`). When `qiflush()` is called, the queues will be flushed when these control characters are read.

The `timeout()` and `wtimeout()` routines set blocking or non-blocking read for a given window. If *delay* is negative, blocking read is used (that is, waits indefinitely for input). If *delay* is zero, then non-blocking read is used (that is, read returns `ERR` if no input is waiting). If *delay* is positive, then read blocks for *delay* milliseconds, and returns `ERR` if there is still no input. Hence, these routines provide the same functionality as `nodelay()`, plus the additional capability of being able to block for only *delay* milliseconds (where *delay* is positive).

`curses` does "line-breakout optimization" by looking for typeahead periodically while updating the screen. If input is found, and it is coming from a tty, the current update is postponed until `refresh()` or `doupdate()` is called again. This allows faster response to commands typed in advance. Normally, the input FILE pointer passed to `newterm()`, or `stdin` in the case that `initscr()` was used, will be used to do this typeahead checking. The `typeahead()` routine specifies that the file descriptor *fildev* is to be used to check for typeahead instead. If *fildev* is `-1`, then no typeahead checking is done.

RETURN VALUES

All routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_getch(3CURSES)`, `curs_initscr(3CURSES)`, `curses(3CURSES)`, `attributes(5)`, `termio(7I)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `echo()`, `noecho()`, `halfdelay()`, `intrflush()`, `meta()`, `nodelay()`, `notimeout()`, `noqiflush()`, `qiflush()`, `timeout()`, and `wtimeout()` may be macros.

NAME halfdelay – enable/disable half-delay mode

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

int halfdelay(int tenths);
```

DESCRIPTION The halfdelay() function is similar to cbreak(3XCURSES) in that when set, characters typed by the user are immediately processed by the program. The difference is that ERR is returned if no input is received after *tenths* tenths seconds.

The nocbreak(3XCURSES) function should be used to leave half-delay mode.

PARAMETERS *tenths* Is the number of tenths of seconds for which to block input (1 to 255).

RETURN VALUES On success, the halfdelay() function returns OK. Otherwise, it returns ERR.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO cbreak(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

has_colors(3CURSES)

| | |
|-----------------------------|--|
| NAME | <code> curs_color, start_color, init_pair, init_color, has_colors, can_change_color, color_content, pair_content</code> – curses color manipulation routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int start_color(void); int init_pair(short <i>pair</i>, short <i>fg</i>, short <i>bg</i>); int init_color(short <i>color</i>, short <i>red</i>, short <i>green</i>, short <i>blue</i>); bool has_colors(void); bool can_change_color(void); int color_content(short <i>color</i>, short *<i>redp</i>, short *<i>greenp</i>, short *<i>bluep</i>); int pair_content(short <i>pair</i>, short *<i>fgp</i>, short *<i>bgp</i>);</pre> |
| Overview | <p>curses provides routines that manipulate color on color alphanumeric terminals. To use these routines <code>start_color()</code> must be called, usually right after <code>initscr()</code>. See <code> curs_initscr(3CURSES)</code>. Colors are always used in pairs (referred to as color-pairs). A color-pair consists of a foreground color (for characters) and a background color (for the field on which the characters are displayed). A programmer initializes a color-pair with the routine <code>init_pair</code>. After it has been initialized, <code>COLOR_PAIR(<i>n</i>)</code>, a macro defined in <code><curses.h></code>, can be used in the same ways other video attributes can be used. If a terminal is capable of redefining colors, the programmer can use the routine <code>init_color()</code> to change the definition of a color. The routines <code>has_colors()</code> and <code>can_change_color()</code> return <code>TRUE</code> or <code>FALSE</code>, depending on whether the terminal has color capabilities and whether the programmer can change the colors. The routine <code>color_content()</code> allows a programmer to identify the amounts of red, green, and blue components in an initialized color. The routine <code>pair_content()</code> allows a programmer to find out how a given color-pair is currently defined.</p> |
| Routine Descriptions | <p>The <code>start_color()</code> routine requires no arguments. It must be called if the programmer wants to use colors, and before any other color manipulation routine is called. It is good practice to call this routine right after <code>initscr()</code>. <code>start_color()</code> initializes eight basic colors (black, red, green, yellow, blue, magenta, cyan, and white), and two global variables, <code>COLORS</code> and <code>COLOR_PAIRS</code> (respectively defining the maximum number of colors and color-pairs the terminal can support). It also restores the colors on the terminal to the values they had when the terminal was just turned on.</p> <p>The <code>init_pair()</code> routine changes the definition of a color-pair. It takes three arguments: the number of the color-pair to be changed, the foreground color number, and the background color number. The value of the first argument must be between 1 and <code>COLOR_PAIRS-1</code>. The value of the second and third arguments must be between 0 and <code>COLORS</code>. If the color-pair was previously initialized, the screen is refreshed and all occurrences of that color-pair is changed to the new definition.</p> |

has_colors(3CURSES)

The `init_color()` routine changes the definition of a color. It takes four arguments: the number of the color to be changed followed by three RGB values (for the amounts of red, green, and blue components). The value of the first argument must be between 0 and `COLORS`. (See the section `COLORS` for the default color index.) Each of the last three arguments must be a value between 0 and 1000. When `init_color()` is used, all occurrences of that color on the screen immediately change to the new definition.

The `has_colors()` routine requires no arguments. It returns `TRUE` if the terminal can manipulate colors; otherwise, it returns `FALSE`. This routine facilitates writing terminal-independent programs. For example, a programmer can use it to decide whether to use color or some other video attribute.

The `can_change_color()` routine requires no arguments. It returns `TRUE` if the terminal supports colors and can change their definitions; other, it returns `FALSE`. This routine facilitates writing terminal-independent programs.

The `color_content()` routine gives users a way to find the intensity of the red, green, and blue (RGB) components in a color. It requires four arguments: the color number, and three addresses of `shorts` for storing the information about the amounts of red, green, and blue components in the given color. The value of the first argument must be between 0 and `COLORS`. The values that are stored at the addresses pointed to by the last three arguments are between 0 (no component) and 1000 (maximum amount of component).

The `pair_content()` routine allows users to find out what colors a given color-pair consists of. It requires three arguments: the color-pair number, and two addresses of `shorts` for storing the foreground and the background color numbers. The value of the first argument must be between 1 and `COLOR_PAIRS-1`. The values that are stored at the addresses pointed to by the second and third arguments are between 0 and `COLORS`.

Colors In `< curses.h >` the following macros are defined. These are the default colors. `curses` also assumes that `COLOR_BLACK` is the default background color for all terminals.

```
COLOR_BLACK
COLOR_RED
COLOR_GREEN
COLOR_YELLOW
COLOR_BLUE
COLOR_MAGENTA
COLOR_CYAN
COLOR_WHITE
```

RETURN VALUES All routines that return an integer return `ERR` upon failure and `OK` upon successful completion.

has_colors(3CURSES)

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_attr(3CURSES), curs_initscr(3CURSES), curses(3CURSES), attributes(5)

NOTES The header < curses . h > automatically includes the headers < stdio . h > and < unctrl . h >.

| | |
|------------------------------|--|
| NAME | can_change_color, color_content, COLOR_PAIR, has_colors, init_color, init_pair, pair_content, PAIR_NUMBER, start_color, COLOR_PAIRS, COLORS – manipulate color information |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> bool can_change_color(void); int color_content(short color, short *red, short *green, short *blue); int COLOR_PAIR(int n); bool has_colors(void); int init_color(short color, short red, short green, short blue); int init_pair(short pair, short f, short b); int pair_content(short pair, short *f, short *b); int PAIR_NUMBER(int value); int start_color(void); extern int COLOR_PAIRS; extern int COLORS;</pre> |
| DESCRIPTION | These functions manipulate color on terminals that support color. |
| Querying Capabilities | The has_colors() function indicates whether the terminal is a color terminal. The can_change_color() function indicates whether the terminal is a color terminal on which colors can be redefined. |
| Initialization | <p>The start_color() function must be called to enable use of colors and before any color manipulation function is called. The function initializes eight basic colors (black, red, green, yellow, blue, magenta, cyan, and white) that can be specified by the color macros (such as COLOR_BLACK) defined in <curses.h>. The initial appearance of these colors is unspecified.</p> <p>The function also initializes two global external variables:</p> <ul style="list-style-type: none"> ■ COLORS defines the number of colors that the terminal supports. See Color Identification below. If COLORS is 0, the terminal does not support redefinition of colors and can_change_color() will return FALSE. ■ COLOR_PAIRS defines the maximum number of color-pairs that the terminal supports. See User-defined Color Pairs below. |

has_colors(3XCURSES)

| | | | | | | | | | | | | | | | |
|---------------------------------|--|--------------|--|------------|---|--------------|---|-------------|--|----------|--------------------------------|-------------|--|----------|---|
| | <p>The <code>start_color()</code> function also restores the colors on the terminal to terminal-specific initial values. The initial background color is assumed to be black for all terminals.</p> | | | | | | | | | | | | | | |
| Color Identification | <p>The <code>init_color()</code> function redefines color number <i>color</i>, on terminals that support the redefinition of colors, to have the red, green, and blue intensity components specified by <i>red</i>, <i>green</i>, and <i>blue</i>, respectively. Calling <code>init_color()</code> also changes all occurrences of the specified color on the screen to the new definition.</p> <p>The <code>color_content()</code> function identifies the intensity components of color number <i>color</i>. It stores the red, green, and blue intensity components of this color in the addresses pointed to by <i>red</i>, <i>green</i>, and <i>blue</i>, respectively.</p> <p>For both functions, the <i>color</i> argument must be in the range from 0 to and including <code>COLORS-1</code>. Valid intensity value range from 0 (no intensity component) up to and including 1000 (maximum intensity in that component).</p> | | | | | | | | | | | | | | |
| User-defined Color Pairs | <p>Calling <code>init_pair()</code> defines or redefines color-pair number <i>pair</i> to have foreground color <i>f</i> and background color <i>b</i>. Calling <code>init_pair()</code> changes any characters that were displayed in the color pair's old definition to the new definition and refreshes the screen.</p> <p>After defining the color pair, the macro <code>COLOR_PAIR(n)</code> returns the value of color pair <i>n</i>. This value is the color attribute as it would be extracted from a <code>chtype</code>. Controversy, the macro <code>COLOR_NUMBER(value)</code> returns the color pair number associated with the color attribute <i>value</i>.</p> <p>The <code>pair_content()</code> retrieves the component colors of a color-pair number <i>pair</i>. It stores the foreground and background color numbers in the variables pointed to by <i>f</i> and <i>b</i>, respectively.</p> <p>With <code>init_pair()</code> and <code>pair_content()</code>, the value of <i>pair</i> must be in a range from 0 to and including <code>COLOR_PAIRS-1</code>. Valid values for <i>f</i> and <i>b</i> are the range from 0 to and including <code>COLORS-1</code>.</p> | | | | | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>color</i></td><td>Is the number of the color for which to provide information (0 to <code>COLORS-1</code>).</td></tr><tr><td><i>red</i></td><td>Is a pointer to the RGB value for the amount of red in <i>color</i>.</td></tr><tr><td><i>green</i></td><td>Is a pointer to the RGB value for the amount of green in <i>color</i>.</td></tr><tr><td><i>blue</i></td><td>Is a pointer to the RGB value for the amount of blue in <i>color</i>.</td></tr><tr><td><i>n</i></td><td>Is the number of a color pair.</td></tr><tr><td><i>pair</i></td><td>Is the number of the color pair for which to provide information (1 to <code>COLOR_PAIRS-1</code>).</td></tr><tr><td><i>f</i></td><td>Is a pointer to the number of the foreground color (0 to <code>COLORS-1</code>) in <i>pair</i>.</td></tr></table> | <i>color</i> | Is the number of the color for which to provide information (0 to <code>COLORS-1</code>). | <i>red</i> | Is a pointer to the RGB value for the amount of red in <i>color</i> . | <i>green</i> | Is a pointer to the RGB value for the amount of green in <i>color</i> . | <i>blue</i> | Is a pointer to the RGB value for the amount of blue in <i>color</i> . | <i>n</i> | Is the number of a color pair. | <i>pair</i> | Is the number of the color pair for which to provide information (1 to <code>COLOR_PAIRS-1</code>). | <i>f</i> | Is a pointer to the number of the foreground color (0 to <code>COLORS-1</code>) in <i>pair</i> . |
| <i>color</i> | Is the number of the color for which to provide information (0 to <code>COLORS-1</code>). | | | | | | | | | | | | | | |
| <i>red</i> | Is a pointer to the RGB value for the amount of red in <i>color</i> . | | | | | | | | | | | | | | |
| <i>green</i> | Is a pointer to the RGB value for the amount of green in <i>color</i> . | | | | | | | | | | | | | | |
| <i>blue</i> | Is a pointer to the RGB value for the amount of blue in <i>color</i> . | | | | | | | | | | | | | | |
| <i>n</i> | Is the number of a color pair. | | | | | | | | | | | | | | |
| <i>pair</i> | Is the number of the color pair for which to provide information (1 to <code>COLOR_PAIRS-1</code>). | | | | | | | | | | | | | | |
| <i>f</i> | Is a pointer to the number of the foreground color (0 to <code>COLORS-1</code>) in <i>pair</i> . | | | | | | | | | | | | | | |

has_colors(3XCURSES)

b Is a pointer to the number of the background color (0 to COLORS-1) in *pair*.

value Is a color attribute value.

RETURN VALUES The has_colors() function returns TRUE if the terminal can manipulate colors. Otherwise, it returns FALSE.

The can_change_color() function returns TRUE if the terminal supports colors and is able to change their definitions. Otherwise, it returns FALSE.

Upon successful completion, the other functions return OK. Otherwise, they return ERR.

ERRORS No errors are defined.

USAGE To use these functions, start_color() must be called, usually right after initscr(3XCURSES).

The can_change_color() and has_colors() functions facilitate writing terminal-independent applications. For example, a programmer can use them to decide whether to use color or some other video attribute.

On color terminals, a typical value of COLORS is 8 and the macros such as COLOR_BLACK return a value within the range from 0 to and including 7. However, applications cannot rely on this to be true.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO attroff(3XCURSES), delscreen(3XCURSES), initscr(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

has_ic(3CURSES)

| | |
|----------------------|---|
| NAME | <code> curs_termattrs, baudrate, erasechar, has_ic, has_il, killchar, longname, termattrs, termname</code> – curses environment query routines |
| SYNOPSIS | <pre>cc [<i>flag ...</i>] <i>file ...</i> -lcurses [<i>library ...</i>] #include <curses.h> int baudrate(void); char erasechar(void); int has_ic(void); int has_il(void); char killchar(void); char *longname(void); chtype termattrs(void); char *termname(void);</pre> |
| DESCRIPTION | <p>The <code>baudrate()</code> routine returns the output speed of the terminal. The number returned is in bits per second, for example 9600, and is an integer.</p> <p>With the <code>erasechar()</code> routine, the user's current erase character is returned.</p> <p>The <code>has_ic()</code> routine is true if the terminal has insert- and delete-character capabilities.</p> <p>The <code>has_il()</code> routine is true if the terminal has insert- and delete-line capabilities, or can simulate them using scrolling regions. This might be used to determine if it would be appropriate to turn on physical scrolling using <code>scrollok()</code>.</p> <p>With the <code>killchar()</code> routine, the user's current line kill character is returned.</p> <p>The <code>longname()</code> routine returns a pointer to a static area containing a verbose description of the current terminal. The maximum length of a verbose description is 128 characters. It is defined only after the call to <code>initscr()</code> or <code>newterm()</code>. The area is overwritten by each call to <code>newterm()</code> and is not restored by <code>set_term()</code>, so the value should be saved between calls to <code>newterm()</code> if <code>longname()</code> is going to be used with multiple terminals.</p> <p>If a given terminal doesn't support a video attribute that an application program is trying to use, <code>curses</code> may substitute a different video attribute for it. The <code>termattrs()</code> function returns a logical OR of all video attributes supported by the terminal. This information is useful when a <code>curses</code> program needs complete control over the appearance of the screen.</p> <p>The <code>termname()</code> routine returns the value of the environment variable <code>TERM</code> (truncated to 14 characters).</p> |
| RETURN VALUES | <code>longname()</code> and <code>termname()</code> return <code>NULL</code> on error. |

has_ic(3CURSES)

Routines that return an integer return ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_initscr(3CURSES), curs_outopts(3CURSES), curses(3CURSES), attributes(5)

NOTES The header < curses . h > automatically includes the headers < stdio . h > and < unctrl . h >.

Note that termattrs () may be a macro.

has_ic(3XCURSES)

NAME | has_ic, has_il – determine insert/delete character/line capability

SYNOPSIS |

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

bool has_ic(void);

bool has_il(void);
```

DESCRIPTION | The `has_ic()` function determines whether or not the terminal has insert/delete character capability.

The `has_il()` function determines whether or not the terminal has insert/delete line capability.

RETURN VALUES | The `has_ic()` function returns `TRUE` if the terminal has insert/delete character capability and `FALSE` otherwise.

The `has_il()` function returns `TRUE` if the terminal has insert/delete line capability and `FALSE` otherwise.

ERRORS | None.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|----------------------|---|
| NAME | curs_termattrs, baudrate, erasechar, has_ic, has_il, killchar, longname, termattrs, termname – curses environment query routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int baudrate(void); char erasechar(void); int has_ic(void); int has_il(void); char killchar(void); char *longname(void); ctype_t termattrs(void); char *termname(void);</pre> |
| DESCRIPTION | <p>The <code>baudrate()</code> routine returns the output speed of the terminal. The number returned is in bits per second, for example 9600, and is an integer.</p> <p>With the <code>erasechar()</code> routine, the user's current erase character is returned.</p> <p>The <code>has_ic()</code> routine is true if the terminal has insert- and delete-character capabilities.</p> <p>The <code>has_il()</code> routine is true if the terminal has insert- and delete-line capabilities, or can simulate them using scrolling regions. This might be used to determine if it would be appropriate to turn on physical scrolling using <code>scrollok()</code>.</p> <p>With the <code>killchar()</code> routine, the user's current line kill character is returned.</p> <p>The <code>longname()</code> routine returns a pointer to a static area containing a verbose description of the current terminal. The maximum length of a verbose description is 128 characters. It is defined only after the call to <code>initscr()</code> or <code>newterm()</code>. The area is overwritten by each call to <code>newterm()</code> and is not restored by <code>set_term()</code>, so the value should be saved between calls to <code>newterm()</code> if <code>longname()</code> is going to be used with multiple terminals.</p> <p>If a given terminal doesn't support a video attribute that an application program is trying to use, <code>curses</code> may substitute a different video attribute for it. The <code>termattrs()</code> function returns a logical OR of all video attributes supported by the terminal. This information is useful when a <code>curses</code> program needs complete control over the appearance of the screen.</p> <p>The <code>termname()</code> routine returns the value of the environment variable <code>TERM</code> (truncated to 14 characters).</p> |
| RETURN VALUES | <code>longname()</code> and <code>termname()</code> return <code>NULL</code> on error. |

has_il(3CURSES)

Routines that return an integer return ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_initscr(3CURSES)`, `curl_outopts(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `termattrs()` may be a macro.

| NAME | has_ic, has_il – determine insert/delete character/line capability | | | | | | |
|----------------------|--|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> bool has_ic(void); bool has_il(void);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>has_ic()</code> function determines whether or not the terminal has insert/delete character capability.</p> <p>The <code>has_il()</code> function determines whether or not the terminal has insert/delete line capability.</p> | | | | | | |
| RETURN VALUES | <p>The <code>has_ic()</code> function returns <code>TRUE</code> if the terminal has insert/delete character capability and <code>FALSE</code> otherwise.</p> <p>The <code>has_il()</code> function returns <code>TRUE</code> if the terminal has insert/delete line capability and <code>FALSE</code> otherwise.</p> | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

hide_panel(3CURSES)

NAME panel_show, show_panel, hide_panel, panel_hidden – panels deck manipulation routines

SYNOPSIS

```
cc [ flag ... ] file ... -lpanel -lcurses [ library .. ]
#include <panel.h>

int show_panel (PANEL *panel) ;
int hide_panel (PANEL *panel) ;
int panel_hidden (PANEL *panel) ;
```

DESCRIPTION show_panel () makes *panel*, previously hidden, visible and places it on top of the deck of panels.

hide_panel () removes *panel* from the panel deck and, thus, hides it from view. The internal data structure of the panel is retained.

panel_hidden () returns TRUE (1) or FALSE (0) indicating whether or not *panel* is in the deck of panels.

RETURN VALUES show_panel () and hide_panel () return the integer OK upon successful completion or ERR upon error.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), panel_update(3CURSES), panels(3CURSES), attributes(5)

NOTES The header <panel.h> automatically includes the header <curses.h>.

| | |
|----------------------|--|
| NAME | hline, mvhline, mvvline, mvwhline, mvwvline, vline, whline, wvline – use single-byte characters (and renditions) to draw lines |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int hline(chtype ch, int n); int mvhline(int y, int x, chtype ch, int n); int mvvline(int y, int x, chtype ch, int n); int mvwhline(WINDOW *win, int y, int x, chtype ch, int n); int mvwvline(WINDOW *win, int y, int x, chtype ch, int n); int vline(chtype ch, int n); int whline(WINDOW *win, chtype ch, int n); int wvline(WINDOW *win, chtype ch, int n);</pre> |
| DESCRIPTION | <p>The <code>hline()</code>, <code>vline()</code>, <code>whline()</code>, <code>wvline()</code> functions draw a horizontal or vertical line, in either the window <code>stdscr</code> or <code>win</code> starting at the current cursor position. The line is drawn using the character <code>ch</code> and is a maximum of <code>n</code> positions long, or as many as will fit into the window. If <code>ch</code> is 0 (zero), the default horizontal or vertical character is used.</p> <p>The <code>mvhline()</code>, <code>mvvline()</code>, <code>mvwhline()</code>, <code>mvwvline()</code> functions are similar to the previous group of functions but the line begins at cursor position specified by <code>x</code> and <code>y</code>.</p> <p>The functions with names ending with <code>hline()</code> draw horizontal lines proceeding towards the last column of the same line. The functions with names ending with <code>vline()</code> draw vertical lines proceeding towards the last column of the same line.</p> <p>These functions do not change the position of the cursor.</p> |
| PARAMETERS | <p><code>ch</code> Is the character used to draw the line.</p> <p><code>n</code> Is the maximum number of characters in the line.</p> <p><code>y</code> Is the y (row) coordinate for the start of the line.</p> <p><code>x</code> Is the x (column) coordinate for the start of the line.</p> <p><code>win</code> Is a pointer to a window.</p> |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. |
| ERRORS | None |

hline(3XCURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `border(3XCURSES)`, `border_set(3XCURSES)`, `hline_set(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|----------------------|---|
| NAME | hline_set, mvhline_set, mvvline_set, mvwhline_set, mvwvline_set, vline_set, whline_set, wvline_set – use complex characters (and renditions) to draw lines |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int hline_set(const cchar_t *ch, int n); int mvhline_set(int y, int x, const cchar_t *wch, int n); int mvvline_set(int y, int x, const cchar_t *wch, int n); int mvwhline_set(WINDOW *win, int y, int x, const cchar_t *wch, int n); int mvwvline_set(WINDOW *win, int y, int x, const cchar_t *wch, int n); int vline_set(const cchar_t *wch, int n); int whline_set(WINDOW *win, const cchar_t *wch, int n); int wvline_set(WINDOW *win, const cchar_t *wch, int n);</pre> |
| DESCRIPTION | <p>The <code>hline_set()</code>, <code>vline_set()</code>, <code>whline_set()</code>, <code>wvline_set()</code> functions draw a line, in either the window <code>stdscr</code> or <code>win</code> starting at the current cursor position. The line is drawn using the character <code>wch</code> and is a maximum of <code>n</code> positions long, or as many as will fit into the window. If <code>wch</code> is a null pointer, the default horizontal or vertical character is used.</p> <p>The <code>mvhline_set()</code>, <code>mvvline_set()</code>, <code>mvwhline_set()</code>, <code>mvwvline_set()</code> functions are similar to the previous group of functions but the line begins at cursor position specified by <code>x</code> and <code>y</code>.</p> <p>The functions with names ending with <code>hline_set()</code> draw horizontal lines proceeding towards the last column of the same line. The functions with names ending with <code>vline_set()</code> draw vertical lines proceeding towards the last column of the same line.</p> <p>These functions do not change the position of the cursor.</p> |
| PARAMETERS | <p><code>wch</code> Is the complex character used to draw the line.</p> <p><code>n</code> Is the maximum number of characters in the line.</p> <p><code>y</code> Is the y (row) coordinate for the start of the line.</p> <p><code>x</code> Is the x (column) coordinate for the start of the line.</p> <p><code>win</code> Is a pointer to a window.</p> |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. |

hline_set(3XCURSES)

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `border(3XCURSES)`, `border_set(3XCURSES)`, `hline(3XCURSES)`,
`libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | curs_ouptoks, clearok, idlok, idcok, immedok, leaveok, setscreg, wsetscreg, scrollok, nl, nonl – curses terminal output option control routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int clearok(WINDOW *win, bool bf); int idlok(WINDOW *win, bool bf); void idcok(WINDOW *win, bool bf); void immedok(WINDOW *win, bool bf); int leaveok(WINDOW *win, bool bf); int setscreg(int top, int bot); int wsetscreg(WINDOW *win, int top, int bot); int scrollok(WINDOW *win, bool bf); int nl(void); int nonl(void);</pre> |
| DESCRIPTION | <p>These routines set options that deal with output within curses. All options are initially FALSE, unless otherwise stated. It is not necessary to turn these options off before calling <code>endwin()</code>.</p> <p>With the <code>clearok()</code> routine, if enabled (<i>bf</i> is TRUE), the next call to <code>wrefresh()</code> with this window will clear the screen completely and redraw the entire screen from scratch. This is useful when the contents of the screen are uncertain, or in some cases for a more pleasing visual effect. If the <i>win</i> argument to <code>clearok()</code> is the global variable <code>curscr()</code>, the next call to <code>wrefresh()</code> with any window causes the screen to be cleared and repainted from scratch.</p> <p>With the <code>idlok()</code> routine, if enabled (<i>bf</i> is TRUE), <code>curses</code> considers using the hardware insert/delete line feature of terminals so equipped. If disabled (<i>bf</i> is FALSE), <code>curses</code> very seldom uses this feature. (The insert/delete character feature is always considered.) This option should be enabled only if the application needs insert/delete line, for example, for a screen editor. It is disabled by default because insert/delete line tends to be visually annoying when used in applications where it isn't really needed. If insert/delete line cannot be used, <code>curses</code> redraws the changed portions of all lines.</p> <p>With the <code>idcok()</code> routine, if enabled (<i>bf</i> is TRUE), <code>curses</code> considers using the hardware insert/delete character feature of terminals so equipped. This is enabled by default.</p> <p>With the <code>immedok()</code> routine, if enabled (<i>bf</i> is TRUE), any change in the window image, such as the ones caused by <code>waddch()</code>, <code>wclrtoeol()</code>, <code>wscrl()</code>, etc., automatically cause a call to <code>wrefresh()</code>. However, it may degrade the performance considerably, due to repeated calls to <code>wrefresh()</code>. It is disabled by default.</p> |

idcok(3CURSES)

Normally, the hardware cursor is left at the location of the window cursor being refreshed. The `leaveok()` option allows the cursor to be left wherever the update happens to leave it. It is useful for applications where the cursor is not used, since it reduces the need for cursor motions. If possible, the cursor is made invisible when this option is enabled.

The `setscrreg()` and `wsetscrreg()` routines allow the application programmer to set a software scrolling region in a window. *top* and *bot* are the line numbers of the top and bottom margin of the scrolling region. (Line 0 is the top line of the window.) If this option and `scrollok()` are enabled, an attempt to move off the bottom margin line causes all lines in the scrolling region to scroll up one line. Only the text of the window is scrolled. (Note that this has nothing to do with the use of a physical scrolling region capability in the terminal, like that in the VT100. If `idlok()` is enabled and the terminal has either a scrolling region or insert/delete line capability, they will probably be used by the output routines.)

The `scrollok()` option controls what happens when the cursor of a window is moved off the edge of the window or scrolling region, either as a result of a newline action on the bottom line, or typing the last character of the last line. If disabled, (*bf* is FALSE), the cursor is left on the bottom line. If enabled, (*bf* is TRUE), `wrefresh()` is called on the window, and the physical terminal and window are scrolled up one line. (Note that in order to get the physical scrolling effect on the terminal, it is also necessary to call `idlok()`.)

The `nl()` and `nonl()` routines control whether newline is translated into carriage return and linefeed on output, and whether return is translated into newline on input. Initially, the translations do occur. By disabling these translations using `nonl()`, `curses` is able to make better use of the linefeed capability, resulting in faster cursor motion.

RETURN VALUES `setscrreg()` and `wsetscrreg()` return OK upon success and ERR upon failure. All other routines that return an integer always return OK.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_addch(3CURSES)`, `curs_clear(3CURSES)`, `curs_initscr(3CURSES)`, `curs_refresh(3CURSES)`, `curs_scroll(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `clearok()`, `leaveok()`, `scrollok()`, `idcok()`, `nl()`, `nonl()`, and `setscrreg()` may be macros.

`idcok(3CURSES)`

The `immedok()` routine is useful for windows that are used as terminal emulators.

idcok(3XCURSES)

NAME | idcok – enable/disable hardware insert-character and delete-character features

SYNOPSIS |

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

void idcok(WINDOW *win, bool bf);
```

DESCRIPTION | The `idcok()` function enables or disables the use of hardware insert-character and delete-character features in *win*. If *bf* is set to `TRUE`, the use of these features in *win* is enabled (if the terminal is equipped). If *bf* is set to `FALSE`, their use in *win* is disabled.

PARAMETERS | *win* Is a pointer to a window.
bf Is a Boolean expression.

RETURN VALUES | The `idcok()` function does not return a value.

ERRORS | None.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | `clearok(3XCURSES)`, `doupdate(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|---|
| NAME | curs_ouptoks, clearok, idlok, idcok, immedok, leaveok, setscreg, wsetscreg, scrollok, nl, nonl – curses terminal output option control routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int clearok(WINDOW *win, bool bf); int idlok(WINDOW *win, bool bf); void idcok(WINDOW *win, bool bf); void immedok(WINDOW *win, bool bf); int leaveok(WINDOW *win, bool bf); int setscreg(int top, int bot); int wsetscreg(WINDOW *win, int top, int bot); int scrollok(WINDOW *win, bool bf); int nl(void); int nonl(void);</pre> |
| DESCRIPTION | <p>These routines set options that deal with output within curses. All options are initially FALSE, unless otherwise stated. It is not necessary to turn these options off before calling <code>endwin()</code>.</p> <p>With the <code>clearok()</code> routine, if enabled (<i>bf</i> is TRUE), the next call to <code>wrefresh()</code> with this window will clear the screen completely and redraw the entire screen from scratch. This is useful when the contents of the screen are uncertain, or in some cases for a more pleasing visual effect. If the <i>win</i> argument to <code>clearok()</code> is the global variable <code>curscr()</code>, the next call to <code>wrefresh()</code> with any window causes the screen to be cleared and repainted from scratch.</p> <p>With the <code>idlok()</code> routine, if enabled (<i>bf</i> is TRUE), <code>curses</code> considers using the hardware insert/delete line feature of terminals so equipped. If disabled (<i>bf</i> is FALSE), <code>curses</code> very seldom uses this feature. (The insert/delete character feature is always considered.) This option should be enabled only if the application needs insert/delete line, for example, for a screen editor. It is disabled by default because insert/delete line tends to be visually annoying when used in applications where it isn't really needed. If insert/delete line cannot be used, <code>curses</code> redraws the changed portions of all lines.</p> <p>With the <code>idcok()</code> routine, if enabled (<i>bf</i> is TRUE), <code>curses</code> considers using the hardware insert/delete character feature of terminals so equipped. This is enabled by default.</p> <p>With the <code>immedok()</code> routine, if enabled (<i>bf</i> is TRUE), any change in the window image, such as the ones caused by <code>waddch()</code>, <code>wclrtoobot()</code>, <code>wscrl()</code>, etc., automatically cause a call to <code>wrefresh()</code>. However, it may degrade the performance considerably, due to repeated calls to <code>wrefresh()</code>. It is disabled by default.</p> |

idlok(3CURSES)

Normally, the hardware cursor is left at the location of the window cursor being refreshed. The `leaveok()` option allows the cursor to be left wherever the update happens to leave it. It is useful for applications where the cursor is not used, since it reduces the need for cursor motions. If possible, the cursor is made invisible when this option is enabled.

The `setscrreg()` and `wsetscrreg()` routines allow the application programmer to set a software scrolling region in a window. *top* and *bot* are the line numbers of the top and bottom margin of the scrolling region. (Line 0 is the top line of the window.) If this option and `scrollok()` are enabled, an attempt to move off the bottom margin line causes all lines in the scrolling region to scroll up one line. Only the text of the window is scrolled. (Note that this has nothing to do with the use of a physical scrolling region capability in the terminal, like that in the VT100. If `idlok()` is enabled and the terminal has either a scrolling region or insert/delete line capability, they will probably be used by the output routines.)

The `scrollok()` option controls what happens when the cursor of a window is moved off the edge of the window or scrolling region, either as a result of a newline action on the bottom line, or typing the last character of the last line. If disabled, (*bf* is FALSE), the cursor is left on the bottom line. If enabled, (*bf* is TRUE), `wrefresh()` is called on the window, and the physical terminal and window are scrolled up one line. (Note that in order to get the physical scrolling effect on the terminal, it is also necessary to call `idlok()`.)

The `nl()` and `nonl()` routines control whether newline is translated into carriage return and linefeed on output, and whether return is translated into newline on input. Initially, the translations do occur. By disabling these translations using `nonl()`, `curses` is able to make better use of the linefeed capability, resulting in faster cursor motion.

RETURN VALUES `setscrreg()` and `wsetscrreg()` return OK upon success and ERR upon failure. All other routines that return an integer always return OK.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_addch(3CURSES)`, `curs_clear(3CURSES)`, `curs_initscr(3CURSES)`, `curs_refresh(3CURSES)`, `curs_scroll(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `clearok()`, `leaveok()`, `scrollok()`, `idcok()`, `nl()`, `nonl()`, and `setscrreg()` may be macros.

`idlok(3CURSES)`

The `immedok()` routine is useful for windows that are used as terminal emulators.

idlok(3XCURSES)

| | |
|--------------------|---|
| NAME | clearok, idlok, leaveok, scrollok, setscrreg, wsetscreg – terminal output control functions |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int clearok(WINDOW *win, bool bf); int idlok(WINDOW *win, bool bf); int leaveok(WINDOW *win, bool bf); int scrollok(WINDOW *win, bool bf); int setscreg(int top, int bot); int wsetscreg(WINDOW *win, int top, int bot);</pre> |
| DESCRIPTION | <p>These functions set options that deal with the output within Curses functions.</p> <p>The <code>clearok()</code> function assigns the value of <i>bf</i> to an internal flag in the specified window that governs clearing of the screen during a refresh. If, during a refresh operation on the specified window, the flag in <i>curscr</i> is TRUE or the flag in the specified window is TRUE, <code>clearok()</code> clears the screen, redraws it in its entirety, and sets the flag to FALSE in <i>curscr</i> and in the specified window. The initial state is unspecified</p> <p>The <code>idlok()</code> function specifies whether the implementation may use the hardware insert-line, delete-line, and scroll features of terminals so equipped. If <i>bf</i> is TRUE, use of these features is enabled. If <i>bf</i> is FALSE, use of these features is disabled and lines are instead redrawn as required. The initial state is FALSE.</p> <p>The <code>leaveok()</code> function controls the cursor position after a refresh operation. If <i>bf</i> is TRUE, refresh operations on the specified window may leave the terminal's cursor at an arbitrary position. If <i>bf</i> is FALSE, then at the end of any refresh operation, the terminal's cursor is positioned at the cursor position contained in the specified window. The initial state is FALSE.</p> <p>The <code>scrollok()</code> function controls the use of scrolling. If <i>bf</i> is TRUE, then scrolling is enabled for the specified window. If <i>bf</i> is FALSE, scrolling is disabled for the specified window. The initial state is FALSE.</p> <p>The <code>setscreg()</code> and <code>wsetscreg()</code> functions define a software scrolling region in the current or specified window. The <i>top</i> and <i>bottom</i> arguments are the line numbers of the first and last line defining the scrolling region. (Line 0 is the top line of the window.) If this option and <code>scrollok()</code> are enabled, an attempt to move off the last line of the margin causes all lines in the scrolling region to scroll one line in the</p> |

idlok(3XCURSES)

direction of the first line. Only characters in the window are scrolled. If a software scrolling region is set and `scrollok()` is not enabled, an attempt to move off the last line of the margin does not reposition any lines in the scrolling region.

PARAMETERS

| | |
|------------|---|
| <i>win</i> | Is a pointer to a window. |
| <i>bf</i> | Is a Boolean expression. |
| <i>top</i> | Is the top line of the scrolling region (top of the window is line 0). |
| <i>bot</i> | Is the bottom line of the scrolling region (top of the window is line 0). |

RETURN VALUES Upon successful completion, the `setscrreg()` and `wsetscrreg()` functions return OK. Otherwise, they return ERR.

The other functions always return OK.

ERRORS No errors are defined.

USAGE The only reason to enable the `idlok()` feature is to use scrolling to achieve the visual effect of motion of a partial window, such as for a screen editor. In other cases, the feature can be visually annoying.

The `leaveok()` option provides greater efficiency for applications that do not use the cursor.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `bkgdset(3XCURSES)`, `clear(3XCURSES)`, `doupdate(3XCURSES)`, `libcurses(3XCURSES)`, `scr1(3XCURSES)`, `attributes(5)`, `standards(5)`

immedok(3CURSES)

| | |
|--------------------|--|
| NAME | <code>curs_ouopts</code> , <code>clearok</code> , <code>idlok</code> , <code>idcok</code> , <code>immedok</code> , <code>leaveok</code> , <code>setscrreg</code> , <code>wsetscrreg</code> , <code>scrollok</code> , <code>nl</code> , <code>nonl</code> – curses terminal output option control routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int clearok(WINDOW *win, bool bf); int idlok(WINDOW *win, bool bf); void idcok(WINDOW *win, bool bf); void immedok(WINDOW *win, bool bf); int leaveok(WINDOW *win, bool bf); int setscrreg(int top, int bot); int wsetscrreg(WINDOW *win, int top, int bot); int scrollok(WINDOW *win, bool bf); int nl(void); int nonl(void);</pre> |
| DESCRIPTION | <p>These routines set options that deal with output within curses. All options are initially FALSE, unless otherwise stated. It is not necessary to turn these options off before calling <code>endwin()</code>.</p> <p>With the <code>clearok()</code> routine, if enabled (<i>bf</i> is TRUE), the next call to <code>wrefresh()</code> with this window will clear the screen completely and redraw the entire screen from scratch. This is useful when the contents of the screen are uncertain, or in some cases for a more pleasing visual effect. If the <i>win</i> argument to <code>clearok()</code> is the global variable <code>curscr()</code>, the next call to <code>wrefresh()</code> with any window causes the screen to be cleared and repainted from scratch.</p> <p>With the <code>idlok()</code> routine, if enabled (<i>bf</i> is TRUE), curses considers using the hardware insert/delete line feature of terminals so equipped. If disabled (<i>bf</i> is FALSE), curses very seldom uses this feature. (The insert/delete character feature is always considered.) This option should be enabled only if the application needs insert/delete line, for example, for a screen editor. It is disabled by default because insert/delete line tends to be visually annoying when used in applications where it isn't really needed. If insert/delete line cannot be used, curses redraws the changed portions of all lines.</p> <p>With the <code>idcok()</code> routine, if enabled (<i>bf</i> is TRUE), curses considers using the hardware insert/delete character feature of terminals so equipped. This is enabled by default.</p> <p>With the <code>immedok()</code> routine, if enabled (<i>bf</i> is TRUE), any change in the window image, such as the ones caused by <code>waddch()</code>, <code>wclrtoeol()</code>, <code>wscrl()</code>, etc., automatically cause a call to <code>wrefresh()</code>. However, it may degrade the performance considerably, due to repeated calls to <code>wrefresh()</code>. It is disabled by default.</p> |

immedok(3CURSES)

Normally, the hardware cursor is left at the location of the window cursor being refreshed. The `leaveok()` option allows the cursor to be left wherever the update happens to leave it. It is useful for applications where the cursor is not used, since it reduces the need for cursor motions. If possible, the cursor is made invisible when this option is enabled.

The `setscrreg()` and `wsetscrreg()` routines allow the application programmer to set a software scrolling region in a window. *top* and *bot* are the line numbers of the top and bottom margin of the scrolling region. (Line 0 is the top line of the window.) If this option and `scrollok()` are enabled, an attempt to move off the bottom margin line causes all lines in the scrolling region to scroll up one line. Only the text of the window is scrolled. (Note that this has nothing to do with the use of a physical scrolling region capability in the terminal, like that in the VT100. If `idlok()` is enabled and the terminal has either a scrolling region or insert/delete line capability, they will probably be used by the output routines.)

The `scrollok()` option controls what happens when the cursor of a window is moved off the edge of the window or scrolling region, either as a result of a newline action on the bottom line, or typing the last character of the last line. If disabled, (*bf* is FALSE), the cursor is left on the bottom line. If enabled, (*bf* is TRUE), `wrefresh()` is called on the window, and the physical terminal and window are scrolled up one line. (Note that in order to get the physical scrolling effect on the terminal, it is also necessary to call `idlok()`.)

The `nl()` and `nonl()` routines control whether newline is translated into carriage return and linefeed on output, and whether return is translated into newline on input. Initially, the translations do occur. By disabling these translations using `nonl()`, `curses` is able to make better use of the linefeed capability, resulting in faster cursor motion.

RETURN VALUES `setscrreg()` and `wsetscrreg()` return OK upon success and ERR upon failure. All other routines that return an integer always return OK.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_addch(3CURSES)`, `curs_clear(3CURSES)`, `curs_initscr(3CURSES)`, `curs_refresh(3CURSES)`, `curs_scroll(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `clearok()`, `leaveok()`, `scrollok()`, `idcok()`, `nl()`, `nonl()`, and `setscrreg()` may be macros.

`immedok(3CURSES)`

The `immedok()` routine is useful for windows that are used as terminal emulators.

NAME immedok – call refresh on changes to window

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

int immedok(WINDOW *win, bool bf);
```

DESCRIPTION If *bf* is TRUE, immedok() calls refresh(3XCURSES) if any change to the window image is made (for example, through functions such as addch(3XCURSES), clrtoobot(3XCURSES), and scl(3XCURSES)). Repeated calls to refresh() may affect performance negatively. The immedok() function is disabled by default.

PARAMETERS *win* Is a pointer to the window that is to be refreshed.
bf Is a Boolean expression.

RETURN VALUES The immedok() function does not return a value.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO addch(3XCURSES), clearok(3XCURSES), clrtoobot(3XCURSES), douupdate(3XCURSES), libcurses(3XCURSES), scl(3XCURSES), attributes(5), standards(5)

inch(3CURSES)

| NAME | <code> curs_inch, inch, winch, mvinch, mvwinch </code> – get a character and its attributes from a curses window | | | | | | |
|---------------------------|--|-------------------------|-------------------------------|---------------------------|--------------------------------|----------------------|--|
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> ctype inch(void); ctype winch(WINDOW *<i>win</i>); ctype mvinch(int <i>y</i>, int <i>x</i>); ctype mvwinch(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>);</pre> | | | | | | |
| DESCRIPTION | With these routines, the character, of type <code>ctype</code> , at the current position in the named window is returned. If any attributes are set for that position, their values are OR-ed into the value returned. Constants defined in <code><curses.h></code> can be used with the logical AND (<code>&</code>) operator to extract the character or attributes alone. | | | | | | |
| Attributes | The following bit-masks can be AND-ed with characters returned by <code>winch()</code> . <table><tr><td><code>A_CHARTEXT</code></td><td>Bit-mask to extract character</td></tr><tr><td><code>A_ATTRIBUTES</code></td><td>Bit-mask to extract attributes</td></tr><tr><td><code>A_COLOR</code></td><td>Bit-mask to extract color-pair field information</td></tr></table> | <code>A_CHARTEXT</code> | Bit-mask to extract character | <code>A_ATTRIBUTES</code> | Bit-mask to extract attributes | <code>A_COLOR</code> | Bit-mask to extract color-pair field information |
| <code>A_CHARTEXT</code> | Bit-mask to extract character | | | | | | |
| <code>A_ATTRIBUTES</code> | Bit-mask to extract attributes | | | | | | |
| <code>A_COLOR</code> | Bit-mask to extract color-pair field information | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . Note that all of these routines may be macros. | | | | | | |

| NAME | inch, mvinch, mvwinch, winch – return a single-byte character (with rendition) | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> ctype inch(void); ctype mvinch(int <i>y</i>, int <i>x</i>); ctype mvwinch(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>); ctype winch(WINDOW *<i>win</i>);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>inch()</code> and <code>winch()</code> functions return the <code>ctype</code> character located at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively. The <code>mvinch()</code> and <code>mvwinch()</code> functions return the <code>ctype</code> character located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The complete character/attribute pair will be returned. The character or attributes can be extracted by performing a bitwise AND on the returned value, using the constants <code>A_CHARTEXT</code>, <code>A_ATTRIBUTES</code>, and <code>A_COLOR</code>.</p> | | | | | | |
| PARAMETERS | <p><i>y</i> Is the <i>y</i> (row) coordinate of the position of the character to be returned.</p> <p><i>x</i> Is the <i>x</i> (column) coordinate of the position of the character to be returned.</p> <p><i>win</i> Is a pointer to the window that contains the character to be returned.</p> | | | | | | |
| RETURN VALUES | On success, these functions return the specified character and rendition. Otherwise, they return <code>ERR</code> . | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>addch(3XCURSES)</code> , <code>attroff(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

inchnstr(3CURSES)

| NAME | <code> curs_inchstr, inchstr, inchnstr, winchstr, winchnstr, mvinchstr, mvwinchnstr, mvwinchnstr</code> – get a string of characters (and attributes) from a curses window | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int inchstr(chtype *chstr); int inchnstr(chtype *chstr, int n); int winchstr(WINDOW *win, chtype *chstr); int winchnstr(WINDOW *win, chtype *chstr, int n); int mvinchstr(int y, int x, chtype *chstr); int mvwinchnstr(int y, int x, chtype *chstr, int n); int mvwinchnstr(WINDOW *win, int y, int x, chtype *chstr); int mvwinchnstr(WINDOW *win, int y, int x, chtype *chstr, int n);</pre> | | | | |
| DESCRIPTION | With these routines, a string of type <code>chtype</code> , starting at the current cursor position in the named window and ending at the right margin of the window, is returned. The four functions with <code>n</code> as the last argument, return the string at most <code>n</code> characters long. Constants defined in <code><curses.h></code> can be used with the <code>&</code> (logical AND) operator to extract the character or the attribute alone from any position in the <code>chstr</code> (see <code>curs_inch(3CURSES)</code>). | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_inch(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . Note that all routines except <code>winchnstr()</code> may be macros. | | | | |

| | |
|--------------------|---|
| NAME | inchnstr, inchstr, mvinchnstr, mvinchstr, mvwinchnstr, mvwinchstr, winchnstr, winchstr – retrieve a single-byte character string (with rendition) |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int inchnstr(chtype *chstr, int n); int inchstr(chtype *chstr); int mvinchnstr(int y, int x, chtype *chstr, int n); int mvinchstr(int y, int x, chtype *chstr); int mvwinchnstr(WINDOW *win, int y, int x, chtype *chstr, int n); int mvwinchstr(WINDOW *win, int y, int x, chtype *chstr); int winchnstr(WINDOW *win, chtype *chstr, int n); int winchstr(WINDOW *win, chtype *chstr);</pre> |
| DESCRIPTION | <p>The <code>inchstr()</code> and <code>winchstr()</code> functions retrieve the character string (with rendition) starting at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively, and ending at the right margin. The <code>mvinchstr()</code> and <code>mvwinchstr()</code> functions retrieve the character string located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>inchnstr()</code>, <code>winchnstr()</code>, <code>mvinchnstr()</code>, and <code>mvwinchnstr()</code> functions retrieve at most <i>n</i> characters from the window <code>stdscr</code> and <i>win</i>, respectively. The former two functions retrieve the string, starting at the current cursor position; the latter two commands retrieve the string, starting at the position specified by the <i>x</i> and <i>y</i> parameters.</p> <p>All these functions store the retrieved character string in the object pointed to by <i>chstr</i>.</p> <p>The complete character/attribute pair is retrieved. The character or attributes can be extracted by performing a bitwise AND on the retrieved value, using the constants <code>A_CHARTEXT</code>, <code>A_ATTRIBUTES</code>, and <code>A_COLOR</code>. The character string can also be retrieved without attributes by using <code>instr(3XCURSES)</code> set of functions.</p> |
| PARAMETERS | <p><i>chstr</i> Is a pointer to an object that can hold the retrieved character string.</p> <p><i>n</i> Is the number of characters not to exceed when retrieving <i>chstr</i>.</p> <p><i>y</i> Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved.</p> <p><i>x</i> Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved.</p> |

inchstr(3XCURSES)

win Is a pointer to the window in which the string is to be retrieved.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `inch(3XCURSES)`, `innstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_inchstr, inchstr, inchnstr, winchstr, winchnstr, mvinchstr, mvwinchnstr, mvwinchstr, mvwinchnstr – get a string of characters (and attributes) from a curses window | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int inchstr(chtype *chstr); int inchnstr(chtype *chstr, int n); int winchstr(WINDOW *win, chtype *chstr); int winchnstr(WINDOW *win, chtype *chstr, int n); int mvinchstr(int y, int x, chtype *chstr); int mvwinchnstr(int y, int x, chtype *chstr, int n); int mvwinchstr(WINDOW *win, int y, int x, chtype *chstr); int mvwinchnstr(WINDOW *win, int y, int x, chtype *chstr, int n);</pre> | | | | |
| DESCRIPTION | With these routines, a string of type chtype, starting at the current cursor position in the named window and ending at the right margin of the window, is returned. The four functions with <i>n</i> as the last argument, return the string at most <i>n</i> characters long. Constants defined in <curses.h> can be used with the & (logical AND) operator to extract the character or the attribute alone from any position in the <i>chstr</i> (see curs_inch(3CURSES)). | | | | |
| RETURN VALUES | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curs_inch(3CURSES), curses(3CURSES), attributes(5) | | | | |
| NOTES | <p>The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.</p> <p>Note that all routines except winchnstr() may be macros.</p> | | | | |

inchstr(3XCURSES)

| | | | | | | | | | |
|--------------------|---|--------------|---|----------|--|----------|--|----------|---|
| NAME | <code>inchstr</code> , <code>inchstr</code> , <code>mvinchnstr</code> , <code>mvinchstr</code> , <code>mvwinchnstr</code> , <code>mvwinchstr</code> , <code>winchnstr</code> , <code>winchstr</code> – retrieve a single-byte character string (with rendition) | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int inchnstr(chtype *chstr, int n); int inchstr(chtype *chstr); int mvinchnstr(int y, int x, chtype *chstr, int n); int mvinchstr(int y, int x, chtype *chstr); int mvwinchnstr(WINDOW *win, int y, int x, chtype *chstr, int n); int mvwinchstr(WINDOW *win, int y, int x, chtype *chstr); int winchnstr(WINDOW *win, chtype *chstr, int n); int winchstr(WINDOW *win, chtype *chstr);</pre> | | | | | | | | |
| DESCRIPTION | <p>The <code>inchstr()</code> and <code>winchstr()</code> functions retrieve the character string (with rendition) starting at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively, and ending at the right margin. The <code>mvinchstr()</code> and <code>mvwinchstr()</code> functions retrieve the character string located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>inchnstr()</code>, <code>winchnstr()</code>, <code>mvinchnstr()</code>, and <code>mvwinchnstr()</code> functions retrieve at most <i>n</i> characters from the window <code>stdscr</code> and <i>win</i>, respectively. The former two functions retrieve the string, starting at the current cursor position; the latter two commands retrieve the string, starting at the position specified by the <i>x</i> and <i>y</i> parameters.</p> <p>All these functions store the retrieved character string in the object pointed to by <i>chstr</i>.</p> <p>The complete character/attribute pair is retrieved. The character or attributes can be extracted by performing a bitwise AND on the retrieved value, using the constants <code>A_CHARTEXT</code>, <code>A_ATTRIBUTES</code>, and <code>A_COLOR</code>. The character string can also be retrieved without attributes by using <code>instr(3XCURSES)</code> set of functions.</p> | | | | | | | | |
| PARAMETERS | <table><tr><td><i>chstr</i></td><td>Is a pointer to an object that can hold the retrieved character string.</td></tr><tr><td><i>n</i></td><td>Is the number of characters not to exceed when retrieving <i>chstr</i>.</td></tr><tr><td><i>y</i></td><td>Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved.</td></tr><tr><td><i>x</i></td><td>Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved.</td></tr></table> | <i>chstr</i> | Is a pointer to an object that can hold the retrieved character string. | <i>n</i> | Is the number of characters not to exceed when retrieving <i>chstr</i> . | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. |
| <i>chstr</i> | Is a pointer to an object that can hold the retrieved character string. | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when retrieving <i>chstr</i> . | | | | | | | | |
| <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | | | | | | | | |
| <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | | | | | | | | |

`inchstr(3XCURSES)`

win Is a pointer to the window in which the string is to be retrieved.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `inch(3XCURSES)`, `innstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

init_color(3CURSES)

| | |
|-----------------------------|---|
| NAME | <code> curs_color, start_color, init_pair, init_color, has_colors, can_change_color, color_content, pair_content</code> – curses color manipulation routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int start_color(void); int init_pair(short pair, short fg, short bg); int init_color(short color, short red, short green, short blue); bool has_colors(void); bool can_change_color(void); int color_content(short color, short *redp, short *greenp, short *bluep); int pair_content(short pair, short *fgp, short *bgp);</pre> |
| Overview | <p>curses provides routines that manipulate color on color alphanumeric terminals. To use these routines <code>start_color()</code> must be called, usually right after <code>initscr()</code>. See <code> curs_initscr(3CURSES)</code>. Colors are always used in pairs (referred to as color-pairs). A color-pair consists of a foreground color (for characters) and a background color (for the field on which the characters are displayed). A programmer initializes a color-pair with the routine <code>init_pair</code>. After it has been initialized, <code>COLOR_PAIR(n)</code>, a macro defined in <code><curses.h></code>, can be used in the same ways other video attributes can be used. If a terminal is capable of redefining colors, the programmer can use the routine <code>init_color()</code> to change the definition of a color. The routines <code>has_colors()</code> and <code>can_change_color()</code> return <code>TRUE</code> or <code>FALSE</code>, depending on whether the terminal has color capabilities and whether the programmer can change the colors. The routine <code>color_content()</code> allows a programmer to identify the amounts of red, green, and blue components in an initialized color. The routine <code>pair_content()</code> allows a programmer to find out how a given color-pair is currently defined.</p> |
| Routine Descriptions | <p>The <code>start_color()</code> routine requires no arguments. It must be called if the programmer wants to use colors, and before any other color manipulation routine is called. It is good practice to call this routine right after <code>initscr()</code>. <code>start_color()</code> initializes eight basic colors (black, red, green, yellow, blue, magenta, cyan, and white), and two global variables, <code>COLORS</code> and <code>COLOR_PAIRS</code> (respectively defining the maximum number of colors and color-pairs the terminal can support). It also restores the colors on the terminal to the values they had when the terminal was just turned on.</p> <p>The <code>init_pair()</code> routine changes the definition of a color-pair. It takes three arguments: the number of the color-pair to be changed, the foreground color number, and the background color number. The value of the first argument must be between 1 and <code>COLOR_PAIRS-1</code>. The value of the second and third arguments must be between 0 and <code>COLORS</code>. If the color-pair was previously initialized, the screen is refreshed and all occurrences of that color-pair is changed to the new definition.</p> |

init_color(3CURSES)

The `init_color()` routine changes the definition of a color. It takes four arguments: the number of the color to be changed followed by three RGB values (for the amounts of red, green, and blue components). The value of the first argument must be between 0 and `COLORS`. (See the section `Colors` for the default color index.) Each of the last three arguments must be a value between 0 and 1000. When `init_color()` is used, all occurrences of that color on the screen immediately change to the new definition.

The `has_colors()` routine requires no arguments. It returns `TRUE` if the terminal can manipulate colors; otherwise, it returns `FALSE`. This routine facilitates writing terminal-independent programs. For example, a programmer can use it to decide whether to use color or some other video attribute.

The `can_change_color()` routine requires no arguments. It returns `TRUE` if the terminal supports colors and can change their definitions; other, it returns `FALSE`. This routine facilitates writing terminal-independent programs.

The `color_content()` routine gives users a way to find the intensity of the red, green, and blue (RGB) components in a color. It requires four arguments: the color number, and three addresses of `shorts` for storing the information about the amounts of red, green, and blue components in the given color. The value of the first argument must be between 0 and `COLORS`. The values that are stored at the addresses pointed to by the last three arguments are between 0 (no component) and 1000 (maximum amount of component).

The `pair_content()` routine allows users to find out what colors a given color-pair consists of. It requires three arguments: the color-pair number, and two addresses of `shorts` for storing the foreground and the background color numbers. The value of the first argument must be between 1 and `COLOR_PAIRS-1`. The values that are stored at the addresses pointed to by the second and third arguments are between 0 and `COLORS`.

Colors In `< curses.h >` the following macros are defined. These are the default colors. `curses` also assumes that `COLOR_BLACK` is the default background color for all terminals.

```
COLOR_BLACK
COLOR_RED
COLOR_GREEN
COLOR_YELLOW
COLOR_BLUE
COLOR_MAGENTA
COLOR_CYAN
COLOR_WHITE
```

RETURN VALUES All routines that return an integer return `ERR` upon failure and `OK` upon successful completion.

init_color(3CURSES)

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_attr(3CURSES), curs_initscr(3CURSES), curses(3CURSES), attributes(5)

NOTES The header `< curses.h >` automatically includes the headers `< stdio.h >` and `< unctrl.h >`.

| | |
|------------------------------|--|
| NAME | can_change_color, color_content, COLOR_PAIR, has_colors, init_color, init_pair, pair_content, PAIR_NUMBER, start_color, COLOR_PAIRS, COLORS – manipulate color information |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> bool can_change_color(void); int color_content(short color, short *red, short *green, short *blue); int COLOR_PAIR(int n); bool has_colors(void); int init_color(short color, short red, short green, short blue); int init_pair(short pair, short f, short b); int pair_content(short pair, short *f, short *b); int PAIR_NUMBER(int value); int start_color(void); extern int COLOR_PAIRS; extern int COLORS;</pre> |
| DESCRIPTION | These functions manipulate color on terminals that support color. |
| Querying Capabilities | The has_colors() function indicates whether the terminal is a color terminal. The can_change_color() function indicates whether the terminal is a color terminal on which colors can be redefined. |
| Initialization | <p>The start_color() function must be called to enable use of colors and before any color manipulation function is called. The function initializes eight basic colors (black, red, green, yellow, blue, magenta, cyan, and white) that can be specified by the color macros (such as COLOR_BLACK) defined in <curses.h>. The initial appearance of these colors is unspecified.</p> <p>The function also initializes two global external variables:</p> <ul style="list-style-type: none"> ■ COLORS defines the number of colors that the terminal supports. See Color Identification below. If COLORS is 0, the terminal does not support redefinition of colors and can_change_color() will return FALSE. ■ COLOR_PAIRS defines the maximum number of color-pairs that the terminal supports. See User-defined Color Pairs below. |

init_color(3XCURSES)

| | | | | | | | | | | | | | | | |
|---------------------------------|--|--------------|--|------------|---|--------------|---|-------------|--|----------|--------------------------------|-------------|--|----------|---|
| | <p>The <code>start_color()</code> function also restores the colors on the terminal to terminal-specific initial values. The initial background color is assumed to be black for all terminals.</p> | | | | | | | | | | | | | | |
| Color Identification | <p>The <code>init_color()</code> function redefines color number <i>color</i>, on terminals that support the redefinition of colors, to have the red, green, and blue intensity components specified by <i>red</i>, <i>green</i>, and <i>blue</i>, respectively. Calling <code>init_color()</code> also changes all occurrences of the specified color on the screen to the new definition.</p> <p>The <code>color_content()</code> function identifies the intensity components of color number <i>color</i>. It stores the red, green, and blue intensity components of this color in the addresses pointed to by <i>red</i>, <i>green</i>, and <i>blue</i>, respectively.</p> <p>For both functions, the <i>color</i> argument must be in the range from 0 to and including <code>COLORS-1</code>. Valid intensity value range from 0 (no intensity component) up to and including 1000 (maximum intensity in that component).</p> | | | | | | | | | | | | | | |
| User-defined Color Pairs | <p>Calling <code>init_pair()</code> defines or redefines color-pair number <i>pair</i> to have foreground color <i>f</i> and background color <i>b</i>. Calling <code>init_pair()</code> changes any characters that were displayed in the color pair's old definition to the new definition and refreshes the screen.</p> <p>After defining the color pair, the macro <code>COLOR_PAIR(n)</code> returns the value of color pair <i>n</i>. This value is the color attribute as it would be extracted from a <code>chtype</code>. Controversy, the macro <code>COLOR_NUMBER(value)</code> returns the color pair number associated with the color attribute <i>value</i>.</p> <p>The <code>pair_content()</code> retrieves the component colors of a color-pair number <i>pair</i>. It stores the foreground and background color numbers in the variables pointed to by <i>f</i> and <i>b</i>, respectively.</p> <p>With <code>init_pair()</code> and <code>pair_content()</code>, the value of <i>pair</i> must be in a range from 0 to and including <code>COLOR_PAIRS-1</code>. Valid values for <i>f</i> and <i>b</i> are the range from 0 to and including <code>COLORS-1</code>.</p> | | | | | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>color</i></td><td>Is the number of the color for which to provide information (0 to <code>COLORS-1</code>).</td></tr><tr><td><i>red</i></td><td>Is a pointer to the RGB value for the amount of red in <i>color</i>.</td></tr><tr><td><i>green</i></td><td>Is a pointer to the RGB value for the amount of green in <i>color</i>.</td></tr><tr><td><i>blue</i></td><td>Is a pointer to the RGB value for the amount of blue in <i>color</i>.</td></tr><tr><td><i>n</i></td><td>Is the number of a color pair.</td></tr><tr><td><i>pair</i></td><td>Is the number of the color pair for which to provide information (1 to <code>COLOR_PAIRS-1</code>).</td></tr><tr><td><i>f</i></td><td>Is a pointer to the number of the foreground color (0 to <code>COLORS-1</code>) in <i>pair</i>.</td></tr></table> | <i>color</i> | Is the number of the color for which to provide information (0 to <code>COLORS-1</code>). | <i>red</i> | Is a pointer to the RGB value for the amount of red in <i>color</i> . | <i>green</i> | Is a pointer to the RGB value for the amount of green in <i>color</i> . | <i>blue</i> | Is a pointer to the RGB value for the amount of blue in <i>color</i> . | <i>n</i> | Is the number of a color pair. | <i>pair</i> | Is the number of the color pair for which to provide information (1 to <code>COLOR_PAIRS-1</code>). | <i>f</i> | Is a pointer to the number of the foreground color (0 to <code>COLORS-1</code>) in <i>pair</i> . |
| <i>color</i> | Is the number of the color for which to provide information (0 to <code>COLORS-1</code>). | | | | | | | | | | | | | | |
| <i>red</i> | Is a pointer to the RGB value for the amount of red in <i>color</i> . | | | | | | | | | | | | | | |
| <i>green</i> | Is a pointer to the RGB value for the amount of green in <i>color</i> . | | | | | | | | | | | | | | |
| <i>blue</i> | Is a pointer to the RGB value for the amount of blue in <i>color</i> . | | | | | | | | | | | | | | |
| <i>n</i> | Is the number of a color pair. | | | | | | | | | | | | | | |
| <i>pair</i> | Is the number of the color pair for which to provide information (1 to <code>COLOR_PAIRS-1</code>). | | | | | | | | | | | | | | |
| <i>f</i> | Is a pointer to the number of the foreground color (0 to <code>COLORS-1</code>) in <i>pair</i> . | | | | | | | | | | | | | | |

init_color(3XCURSES)

b Is a pointer to the number of the background color (0 to COLORS-1) in *pair*.

value Is a color attribute value.

RETURN VALUES The `has_colors()` function returns `TRUE` if the terminal can manipulate colors. Otherwise, it returns `FALSE`.

The `can_change_color()` function returns `TRUE` if the terminal supports colors and is able to change their definitions. Otherwise, it returns `FALSE`.

Upon successful completion, the other functions return `OK`. Otherwise, they return `ERR`.

ERRORS No errors are defined.

USAGE To use these functions, `start_color()` must be called, usually right after `initscr(3XCURSES)`.

The `can_change_color()` and `has_colors()` functions facilitate writing terminal-independent applications. For example, a programmer can use them to decide whether to use color or some other video attribute.

On color terminals, a typical value of `COLORS` is 8 and the macros such as `COLOR_BLACK` return a value within the range from 0 to and including 7. However, applications cannot rely on this to be true.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attroff(3XCURSES)`, `delscreen(3XCURSES)`, `initscr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

init_pair(3CURSES)

| | |
|-----------------------------|---|
| NAME | <code>curs_color</code> , <code>start_color</code> , <code>init_pair</code> , <code>init_color</code> , <code>has_colors</code> , <code>can_change_color</code> , <code>color_content</code> , <code>pair_content</code> – curses color manipulation routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int start_color(void); int init_pair(short <i>pair</i>, short <i>fg</i>, short <i>bg</i>); int init_color(short <i>color</i>, short <i>red</i>, short <i>green</i>, short <i>blue</i>); bool has_colors(void); bool can_change_color(void); int color_content(short <i>color</i>, short *<i>redp</i>, short *<i>greenp</i>, short *<i>bluep</i>); int pair_content(short <i>pair</i>, short *<i>fgp</i>, short *<i>bgp</i>);</pre> |
| Overview | <p>curses provides routines that manipulate color on color alphanumeric terminals. To use these routines <code>start_color()</code> must be called, usually right after <code>initscr()</code>. See <code>curs_initscr(3CURSES)</code>. Colors are always used in pairs (referred to as color-pairs). A color-pair consists of a foreground color (for characters) and a background color (for the field on which the characters are displayed). A programmer initializes a color-pair with the routine <code>init_pair</code>. After it has been initialized, <code>COLOR_PAIR(<i>n</i>)</code>, a macro defined in <code><curses.h></code>, can be used in the same ways other video attributes can be used. If a terminal is capable of redefining colors, the programmer can use the routine <code>init_color()</code> to change the definition of a color. The routines <code>has_colors()</code> and <code>can_change_color()</code> return <code>TRUE</code> or <code>FALSE</code>, depending on whether the terminal has color capabilities and whether the programmer can change the colors. The routine <code>color_content()</code> allows a programmer to identify the amounts of red, green, and blue components in an initialized color. The routine <code>pair_content()</code> allows a programmer to find out how a given color-pair is currently defined.</p> |
| Routine Descriptions | <p>The <code>start_color()</code> routine requires no arguments. It must be called if the programmer wants to use colors, and before any other color manipulation routine is called. It is good practice to call this routine right after <code>initscr()</code>. <code>start_color()</code> initializes eight basic colors (black, red, green, yellow, blue, magenta, cyan, and white), and two global variables, <code>COLORS</code> and <code>COLOR_PAIRS</code> (respectively defining the maximum number of colors and color-pairs the terminal can support). It also restores the colors on the terminal to the values they had when the terminal was just turned on.</p> <p>The <code>init_pair()</code> routine changes the definition of a color-pair. It takes three arguments: the number of the color-pair to be changed, the foreground color number, and the background color number. The value of the first argument must be between 1 and <code>COLOR_PAIRS-1</code>. The value of the second and third arguments must be between 0 and <code>COLORS</code>. If the color-pair was previously initialized, the screen is refreshed and all occurrences of that color-pair is changed to the new definition.</p> |

init_pair(3CURSES)

The `init_color()` routine changes the definition of a color. It takes four arguments: the number of the color to be changed followed by three RGB values (for the amounts of red, green, and blue components). The value of the first argument must be between 0 and `COLORS`. (See the section `COLORS` for the default color index.) Each of the last three arguments must be a value between 0 and 1000. When `init_color()` is used, all occurrences of that color on the screen immediately change to the new definition.

The `has_colors()` routine requires no arguments. It returns `TRUE` if the terminal can manipulate colors; otherwise, it returns `FALSE`. This routine facilitates writing terminal-independent programs. For example, a programmer can use it to decide whether to use color or some other video attribute.

The `can_change_color()` routine requires no arguments. It returns `TRUE` if the terminal supports colors and can change their definitions; other, it returns `FALSE`. This routine facilitates writing terminal-independent programs.

The `color_content()` routine gives users a way to find the intensity of the red, green, and blue (RGB) components in a color. It requires four arguments: the color number, and three addresses of `shorts` for storing the information about the amounts of red, green, and blue components in the given color. The value of the first argument must be between 0 and `COLORS`. The values that are stored at the addresses pointed to by the last three arguments are between 0 (no component) and 1000 (maximum amount of component).

The `pair_content()` routine allows users to find out what colors a given color-pair consists of. It requires three arguments: the color-pair number, and two addresses of `shorts` for storing the foreground and the background color numbers. The value of the first argument must be between 1 and `COLOR_PAIRS-1`. The values that are stored at the addresses pointed to by the second and third arguments are between 0 and `COLORS`.

Colors In `< curses.h >` the following macros are defined. These are the default colors. `curses` also assumes that `COLOR_BLACK` is the default background color for all terminals.

```
COLOR_BLACK
COLOR_RED
COLOR_GREEN
COLOR_YELLOW
COLOR_BLUE
COLOR_MAGENTA
COLOR_CYAN
COLOR_WHITE
```

RETURN VALUES All routines that return an integer return `ERR` upon failure and `OK` upon successful completion.

init_pair(3CURSES)

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_attr(3CURSES), curs_initscr(3CURSES), curses(3CURSES), attributes(5)

NOTES The header < curses . h > automatically includes the headers < stdio . h > and < unctrl . h >.

| | |
|------------------------------|--|
| NAME | can_change_color, color_content, COLOR_PAIR, has_colors, init_color, init_pair, pair_content, PAIR_NUMBER, start_color, COLOR_PAIRS, COLORS – manipulate color information |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> bool can_change_color(void); int color_content(short color, short *red, short *green, short *blue); int COLOR_PAIR(int n); bool has_colors(void); int init_color(short color, short red, short green, short blue); int init_pair(short pair, short f, short b); int pair_content(short pair, short *f, short *b); int PAIR_NUMBER(int value); int start_color(void); extern int COLOR_PAIRS; extern int COLORS;</pre> |
| DESCRIPTION | These functions manipulate color on terminals that support color. |
| Querying Capabilities | The has_colors() function indicates whether the terminal is a color terminal. The can_change_color() function indicates whether the terminal is a color terminal on which colors can be redefined. |
| Initialization | <p>The start_color() function must be called to enable use of colors and before any color manipulation function is called. The function initializes eight basic colors (black, red, green, yellow, blue, magenta, cyan, and white) that can be specified by the color macros (such as COLOR_BLACK) defined in <curses.h>. The initial appearance of these colors is unspecified.</p> <p>The function also initializes two global external variables:</p> <ul style="list-style-type: none"> ■ COLORS defines the number of colors that the terminal supports. See Color Identification below. If COLORS is 0, the terminal does not support redefinition of colors and can_change_color() will return FALSE. ■ COLOR_PAIRS defines the maximum number of color-pairs that the terminal supports. See User-defined Color Pairs below. |

init_pair(3XCURSES)

| | | | | | | | | | | | | | | | |
|---------------------------------|--|--------------|--|------------|---|--------------|---|-------------|--|----------|--------------------------------|-------------|--|----------|---|
| | <p>The <code>start_color()</code> function also restores the colors on the terminal to terminal-specific initial values. The initial background color is assumed to be black for all terminals.</p> | | | | | | | | | | | | | | |
| Color Identification | <p>The <code>init_color()</code> function redefines color number <i>color</i>, on terminals that support the redefinition of colors, to have the red, green, and blue intensity components specified by <i>red</i>, <i>green</i>, and <i>blue</i>, respectively. Calling <code>init_color()</code> also changes all occurrences of the specified color on the screen to the new definition.</p> <p>The <code>color_content()</code> function identifies the intensity components of color number <i>color</i>. It stores the red, green, and blue intensity components of this color in the addresses pointed to by <i>red</i>, <i>green</i>, and <i>blue</i>, respectively.</p> <p>For both functions, the <i>color</i> argument must be in the range from 0 to and including <code>COLORS-1</code>. Valid intensity value range from 0 (no intensity component) up to and including 1000 (maximum intensity in that component).</p> | | | | | | | | | | | | | | |
| User-defined Color Pairs | <p>Calling <code>init_pair()</code> defines or redefines color-pair number <i>pair</i> to have foreground color <i>f</i> and background color <i>b</i>. Calling <code>init_pair()</code> changes any characters that were displayed in the color pair's old definition to the new definition and refreshes the screen.</p> <p>After defining the color pair, the macro <code>COLOR_PAIR(<i>n</i>)</code> returns the value of color pair <i>n</i>. This value is the color attribute as it would be extracted from a <code>chtype</code>. Controversy, the macro <code>COLOR_NUMBER(<i>value</i>)</code> returns the color pair number associated with the color attribute <i>value</i>.</p> <p>The <code>pair_content()</code> retrieves the component colors of a color-pair number <i>pair</i>. It stores the foreground and background color numbers in the variables pointed to by <i>f</i> and <i>b</i>, respectively.</p> <p>With <code>init_pair()</code> and <code>pair_content()</code>, the value of <i>pair</i> must be in a range from 0 to and including <code>COLOR_PAIRS-1</code>. Valid values for <i>f</i> and <i>b</i> are the range from 0 to and including <code>COLORS-1</code>.</p> | | | | | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>color</i></td><td>Is the number of the color for which to provide information (0 to <code>COLORS-1</code>).</td></tr><tr><td><i>red</i></td><td>Is a pointer to the RGB value for the amount of red in <i>color</i>.</td></tr><tr><td><i>green</i></td><td>Is a pointer to the RGB value for the amount of green in <i>color</i>.</td></tr><tr><td><i>blue</i></td><td>Is a pointer to the RGB value for the amount of blue in <i>color</i>.</td></tr><tr><td><i>n</i></td><td>Is the number of a color pair.</td></tr><tr><td><i>pair</i></td><td>Is the number of the color pair for which to provide information (1 to <code>COLOR_PAIRS-1</code>).</td></tr><tr><td><i>f</i></td><td>Is a pointer to the number of the foreground color (0 to <code>COLORS-1</code>) in <i>pair</i>.</td></tr></table> | <i>color</i> | Is the number of the color for which to provide information (0 to <code>COLORS-1</code>). | <i>red</i> | Is a pointer to the RGB value for the amount of red in <i>color</i> . | <i>green</i> | Is a pointer to the RGB value for the amount of green in <i>color</i> . | <i>blue</i> | Is a pointer to the RGB value for the amount of blue in <i>color</i> . | <i>n</i> | Is the number of a color pair. | <i>pair</i> | Is the number of the color pair for which to provide information (1 to <code>COLOR_PAIRS-1</code>). | <i>f</i> | Is a pointer to the number of the foreground color (0 to <code>COLORS-1</code>) in <i>pair</i> . |
| <i>color</i> | Is the number of the color for which to provide information (0 to <code>COLORS-1</code>). | | | | | | | | | | | | | | |
| <i>red</i> | Is a pointer to the RGB value for the amount of red in <i>color</i> . | | | | | | | | | | | | | | |
| <i>green</i> | Is a pointer to the RGB value for the amount of green in <i>color</i> . | | | | | | | | | | | | | | |
| <i>blue</i> | Is a pointer to the RGB value for the amount of blue in <i>color</i> . | | | | | | | | | | | | | | |
| <i>n</i> | Is the number of a color pair. | | | | | | | | | | | | | | |
| <i>pair</i> | Is the number of the color pair for which to provide information (1 to <code>COLOR_PAIRS-1</code>). | | | | | | | | | | | | | | |
| <i>f</i> | Is a pointer to the number of the foreground color (0 to <code>COLORS-1</code>) in <i>pair</i> . | | | | | | | | | | | | | | |

init_pair(3XCURSES)

b Is a pointer to the number of the background color (0 to COLORS-1) in *pair*.

value Is a color attribute value.

RETURN VALUES The `has_colors()` function returns `TRUE` if the terminal can manipulate colors. Otherwise, it returns `FALSE`.

The `can_change_color()` function returns `TRUE` if the terminal supports colors and is able to change their definitions. Otherwise, it returns `FALSE`.

Upon successful completion, the other functions return `OK`. Otherwise, they return `ERR`.

ERRORS No errors are defined.

USAGE To use these functions, `start_color()` must be called, usually right after `initscr(3XCURSES)`.

The `can_change_color()` and `has_colors()` functions facilitate writing terminal-independent applications. For example, a programmer can use them to decide whether to use color or some other video attribute.

On color terminals, a typical value of `COLORS` is 8 and the macros such as `COLOR_BLACK` return a value within the range from 0 to and including 7. However, applications cannot rely on this to be true.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attroff(3XCURSES)`, `delscreen(3XCURSES)`, `initscr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

initscr(3CURSES)

| | |
|--------------------|--|
| NAME | <code>curs_initscr</code> , <code>initscr</code> , <code>newterm</code> , <code>endwin</code> , <code>isendwin</code> , <code>set_term</code> , <code>delscreen</code> – curses screen initialization and manipulation routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> WINDOW *initscr(void); int endwin(void); int isendwin(void); SCREEN *newterm(char *type, FILE *outfd, FILE *infd); SCREEN *set_term(SCREEN *new); void delscreen(SCREEN *sp);</pre> |
| DESCRIPTION | <p><code>initscr()</code> is almost always the first routine that should be called (the exceptions are <code>slk_init()</code>, <code>filter()</code>, <code>ripoffline()</code>, <code>use_env()</code> and, for multiple-terminal applications, <code>newterm()</code>.) This determines the terminal type and initializes all curses data structures. <code>initscr()</code> also causes the first call to <code>refresh()</code> to clear the screen. If errors occur, <code>initscr()</code> writes an appropriate error message to standard error and exits; otherwise, a pointer is returned to <code>stdscr()</code>. If the program needs an indication of error conditions, <code>newterm()</code> should be used instead of <code>initscr()</code>; <code>initscr()</code> should only be called once per application.</p> <p>A program that outputs to more than one terminal should use the <code>newterm()</code> routine for each terminal instead of <code>initscr()</code>. A program that needs an indication of error conditions, so it can continue to run in a line-oriented mode if the terminal cannot support a screen-oriented program, would also use this routine. The routine <code>newterm()</code> should be called once for each terminal. It returns a variable of type <code>SCREEN *</code> which should be saved as a reference to that terminal. The arguments are the <i>type</i> of the terminal to be used in place of <code>\$TERM</code>, a file pointer for output to the terminal, and another file pointer for input from the terminal (if <i>type</i> is <code>NULL</code>, <code>\$TERM</code> will be used). The program must also call <code>endwin()</code> for each terminal being used before exiting from curses. If <code>newterm()</code> is called more than once for the same terminal, the first terminal referred to must be the last one for which <code>endwin()</code> is called.</p> <p>A program should always call <code>endwin()</code> before exiting or escaping from curses mode temporarily. This routine restores tty modes, moves the cursor to the lower left-hand corner of the screen and resets the terminal into the proper non-visual mode. Calling <code>refresh()</code> or <code>doupdate()</code> after a temporary escape causes the program to resume visual mode.</p> <p>The <code>isendwin()</code> routine returns <code>TRUE</code> if <code>endwin()</code> has been called without any subsequent calls to <code>wrefresh()</code>, and <code>FALSE</code> otherwise.</p> |

initscr(3CURSES)

The `set_term()` routine is used to switch between different terminals. The screen reference `new` becomes the new current terminal. The previous terminal is returned by the routine. This is the only routine which manipulates `SCREEN` pointers; all other routines affect only the current terminal.

The `delscreen()` routine frees storage associated with the `SCREEN` data structure. The `endwin()` routine does not do this, so `delscreen()` should be called after `endwin()` if a particular `SCREEN` is no longer needed.

RETURN VALUES `endwin()` returns the integer `ERR` upon failure and `OK` upon successful completion.

Routines that return pointers always return `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_kernel(3CURSES)`, `curl_refresh(3CURSES)`, `curl_slk(3CURSES)`, `curl_util(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `initscr()` and `newterm()` may be macros.

initscr(3XCURSES)

NAME initscr, newterm – screen initialization functions

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

WINDOW *initscr(void);

SCREEN *newterm(char *type, FILE *outfp, FILE *infp);
```

PARAMETERS

type Is a string defining the terminal type to be used in place of TERM.

outfp Is a pointer to a file to be used for output to the terminal.

infp Is the pointer to a file to be used for input to the terminal.

DESCRIPTION

The `initscr()` function initializes X/Open Curses data structures, determines the terminal type, and ensures the first call to `refresh(3XCURSES)` clears the screen.

The `newterm()` function opens a new terminal with each call. It should be used instead of `initscr()` when the program interacts with more than one terminal. It returns a variable of type `SCREEN`, which should be used for later reference to that terminal. Before program termination, `endwin()` should be called for each terminal.

The only functions that you can call before calling `initscr()` or `newterm()` are `filter(3XCURSES)`, `riporffline(3XCURSES)`, `slk_init(3XCURSES)`, and `use_env(3XCURSES)`.

RETURN VALUES

On success, the `initscr()` function returns a pointer to `stdscr`; otherwise, `initscr()` does not return.

On success, the `newterm()` function returns a pointer to the specified terminal; otherwise, a null pointer is returned.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

initscr(3XCURSES)

SEE ALSO del_curterm(3XCURSES), delscreen(3XCURSES), douupdate(3XCURSES),
endwin(3XCURSES), filter(3XCURSES), libcurses(3XCURSES),
slk_attroff(3XCURSES), use_env(3XCURSES), attributes(5), standards(5)

innstr(3CURSES)

| NAME | <code> curs_instr, instr, innstr, winstr, winnstr, mvinstr, mvinnstr, mvwinstr, mvwinnstr</code> – get a string of characters from a curses window | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int instr(char *str); int innstr(char *str, int n); int winstr(WINDOW *win, char *str); int winnstr(WINDOW *win, char *str, int n); int mvinstr(int y, int x, char *str); int mvinnstr(int y, int x, char *str, int n); int mvwinstr(WINDOW *win, int y, int x, char *str); int mvwinnstr(WINDOW *win, int y, int x, char *str, int n);</pre> | | | | |
| DESCRIPTION | These routines return a string of characters in <i>str</i> , starting at the current cursor position in the named window and ending at the right margin of the window. Attributes are stripped from the characters. The four functions with <i>n</i> as the last argument return the string at most <i>n</i> characters long. | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . Note that all routines except <code>winnstr()</code> may be macros. | | | | |

| | | | | | | | | | | | |
|--------------------|---|------------|---|----------|--|----------|--|----------|---|------------|--|
| NAME | innstr, instr, mvinnstr, mvinstr, mvwinstr, mvwinstr, winstr – retrieve a multibyte character string (without rendition) | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int innstr(char *str, int n); int instr(char *str); int mvinnstr(int y, int x, char *str, int n); int mvinstr(int y, int x, char *str); int mvwinstr(WINDOW *win, int y, int x, char *str, int n); int mvwinstr(WINDOW *win, int y, int x, char *str); int winstr(WINDOW *win, char *str); int winstr(WINDOW *win, char *str, int n);</pre> | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>str</i></td> <td>Is a pointer to an object that can hold the retrieved multibyte character string.</td> </tr> <tr> <td><i>n</i></td> <td>Is the number of characters not to exceed when retrieving <i>str</i>.</td> </tr> <tr> <td><i>y</i></td> <td>Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved.</td> </tr> <tr> <td><i>x</i></td> <td>Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved.</td> </tr> <tr> <td><i>win</i></td> <td>Is a pointer to the window in which the string is to be retrieved.</td> </tr> </table> | <i>str</i> | Is a pointer to an object that can hold the retrieved multibyte character string. | <i>n</i> | Is the number of characters not to exceed when retrieving <i>str</i> . | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | <i>win</i> | Is a pointer to the window in which the string is to be retrieved. |
| <i>str</i> | Is a pointer to an object that can hold the retrieved multibyte character string. | | | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when retrieving <i>str</i> . | | | | | | | | | | |
| <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | | | | | | | | | | |
| <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the string is to be retrieved. | | | | | | | | | | |
| DESCRIPTION | <p>The <code>instr()</code> and <code>winstr()</code> functions retrieve a multibyte character string (without attributes) starting at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively, and ending at the right margin. The <code>mvinstr()</code> and <code>mvwinstr()</code> functions retrieve a multibyte character string located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>innstr()</code>, <code>winnstr()</code>, <code>mvinnstr()</code>, and <code>mvwinstr()</code> functions retrieve at most <i>n</i> characters from the window <code>stdscr</code> and <i>win</i>, respectively. The former two functions retrieve the string starting at the current cursor position; the latter two commands return the string, starting at the position specified by the <i>x</i> and <i>y</i> parameters.</p> | | | | | | | | | | |

innstr(3XCURSES)

All these functions store the retrieved string in the object pointed to by *str*. They only store complete multibyte characters. If the area pointed to by *str* is not large enough to hold at least one character, these functions fail.

Only the character portion of the character/rendition pair is returned. To return the complete character/rendition pair, use `winchstr()`.

ERRORS OK Successful completion.

ERR An error occurred.

USAGE All functions except `winnstr()` may be macros.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `inch(3XCURSES)`, `inchstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

NAME curs_inwstr, inwstr, innwstr, winwstr, winnwstr, mvinwstr, mvinnwstr, mvwinwstr, mvwinnwstr – get a string of wchar_t characters from a curses window

SYNOPSIS

```
cc [ flag ... ] file ... -lcurses[library .. ]
#include <curses.h>

int inwstr(wchar_t *wstr);
int innwstr(wchar_t *wstr, int n);
int winwstr(WINDOW *win, wchar_t *wstr);
int winnwstr(WINDOW *win, wchar_t *wstr, int n);
int mvinwstr(int y, int x, wchar_t *wstr);
int mvinnwstr(int y, int x, wchar_t *wstr, int n);
int mvwinwstr(WINDOW *win, int y, int x, wchar_t *wstr);
int mvwinnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);
```

DESCRIPTION These routines return the string of wchar_t characters in *wstr* starting at the current cursor position in the named window and ending at the right margin of the window. Attributes are stripped from the characters. The four functions with *n* as the last argument return the string at most *n* wchar_t characters long.

RETURN VALUES All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), attributes(5)

NOTES The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <wider.h>.

Note that all routines except winnwstr() may be macros.

innwstr(3XCURSES)

| | |
|--------------------|---|
| NAME | <code>innwstr</code> , <code>inwstr</code> , <code>mvinnwstr</code> , <code>mvinwstr</code> , <code>mvwinnwstr</code> , <code>mvwinwstr</code> , <code>winnwstr</code> , <code>winwstr</code> – retrieve a wide character string (without rendition) |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int innwstr(wchar_t *wstr, int n); int inwstr(wchar_t *wstr); int mvinnwstr(int y, int x, wchar_t *wstr, int n); int mvinwstr(int y, int x, wchar_t *wstr); int mvwinnwstr(WINDOW*win, int y, int x, wchar_t *wstr, int n); int mvwinwstr(WINDOW*win, int y, int x, wchar_t *wstr); int winwstr(WINDOW*win, wchar_t *wstr); int winnwstr(WINDOW*win, wchar_t *wstr, int n);</pre> |
| PARAMETERS | <p><i>wstr</i> Is a pointer to an object that can hold the retrieved multibyte character string.</p> <p><i>n</i> Is the number of characters not to exceed when retrieving <i>wstr</i>.</p> <p><i>y</i> Is the y (row) coordinate of the starting position of the string to be retrieved.</p> <p><i>x</i> Is the x (column) coordinate of the starting position of the string to be retrieved.</p> <p><i>win</i> Is a pointer to the window in which the string is to be retrieved.</p> |
| DESCRIPTION | <p>The <code>inwstr()</code> and <code>winwstr()</code> functions retrieve a wide character string (without attributes) starting at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively, and ending at the right margin. The <code>mvinwstr()</code> and <code>mvwinwstr()</code> functions retrieve a wide character string located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>innwstr()</code>, <code>winnwstr()</code>, <code>mvinnwstr()</code>, and <code>mvwinnwstr()</code> functions retrieve at most <i>n</i> characters from the window <code>stdscr</code> and <i>win</i>, respectively. The former two functions retrieve the string starting at the current cursor position; the latter two commands return the string, starting at the position specified by the <i>x</i> and <i>y</i> parameters.</p> <p>All these functions store the retrieved string in the object pointed to by <i>wstr</i>. They only store complete wide characters. If the area pointed to by <i>wstr</i> is not large enough to hold at least one character, these functions fail.</p> |

innwstr(3XCURSES)

Only the character portion of the character/rendition pair is returned. To return the complete character/rendition pair, use `win_wchstr(3XCURSES)`.

RETURN VALUES

On success, the `inwstr()`, `mvinwstr()`, `mvwinwstr()`, and `winwstr()` functions return OK. Otherwise, they return ERR.

On success, the `innwstr()`, `mvinnwstr()`, `mvwinnwstr()`, and `winnwstr()` functions return the number of characters read into the string. Otherwise, they return ERR.

ERRORS

None.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO

`in_wch(3XCURSES)`, `in_wchnstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

insch(3CURSES)

- NAME** curs_insch, insch, winsch, mvwinsch, mvwinsch – insert a character before the character under the cursor in a curses window
- SYNOPSIS**

```
cc [ flag ... ] file... -lcurses [ library ... ]
#include <curses.h>

int insch(chtype ch);
int winsch(WINDOW *win, chtype ch);
int mvinsch(int y, int x, chtype ch);
int mvwinsch(WINDOW *win, int y, int x, chtype ch);
```
- DESCRIPTION** With these routines, the character *ch* is inserted before the character under the cursor. All characters to the right of the cursor are moved one space to the right, with the possibility of the rightmost character on the line being lost. The cursor position does not change (after moving to *y*, *x*, if specified). (This does not imply use of the hardware insert character feature.)
- RETURN VALUES** All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.
- ATTRIBUTES** See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

- SEE ALSO** curses(3CURSES), attributes(5)
- NOTES** The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.
- Note that `insch()`, `mvinsch()`, and `mvwinsch()` may be macros.

NAME insch, winsch, mvinsch, mvwinsch – insert a character

SYNOPSIS `cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \`
`-R /usr/xpg4/lib -lcurses [library...]`

`c89 [flag...] file... -lcurses [library...]`

```
#include <curses.h>
int insch(chtype ch);
int mvinsch(int y, int x, chtype ch);
int mvwinsch(WINDOW *win, int y, int x, chtype ch);
int winsch(WINDOW *win, chtype ch);
```

PARAMETERS

| | |
|------------|--|
| <i>ch</i> | Is the character to be inserted. |
| <i>y</i> | Is the y (row) coordinate of the position of the character. |
| <i>x</i> | Is the x (column) coordinate of the position of the character. |
| <i>win</i> | Is a pointer to the window in which the character is to be inserted. |

DESCRIPTION These functions insert the character and rendition from *ch* into the current or specified window at the current or specified position.

These functions do not perform wrapping and do not advance the cursor position. These functions perform special-character processing, with the exception that if a newline is inserted into the last line of a window and scrolling is not enabled, the behavior is unspecified.

RETURN VALUES Upon successful completion, these functions return OK. Otherwise, they return ERR.

ERRORS No errors are defined.

USAGE These functions are only guaranteed to operate reliably on character sets in which each character fits into a single byte, whose attributes can be expressed using only constants with the `A_` prefix.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `ins_wch(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

insdelln(3CURSES)

| NAME | <code> curs_deleteln, deleteln, wdeleteln, insdelln, winsdelln, insertln, winsertln</code> – delete and insert lines in a curses window | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int deleteln(void); int wdeleteln(WINDOW *<i>win</i>); int insdelln(int <i>n</i>); int winsdelln(WINDOW *<i>win</i>, int <i>n</i>); int insertln(void); int winsertln(WINDOW *<i>win</i>);</pre> | | | | |
| DESCRIPTION | <p>With the <code>deleteln()</code> and <code>wdeleteln()</code> routines, the line under the cursor in the window is deleted; all lines below the current line are moved up one line. The bottom line of the window is cleared. The cursor position does not change. This does not imply use of a hardware delete line feature.</p> <p>With the <code>insdelln()</code> and <code>winsdelln()</code> routines, for positive <i>n</i>, insert <i>n</i> lines into the specified window above the current line. The <i>n</i> bottom lines are lost. For negative <i>n</i>, delete <i>n</i> lines (starting with the one under the cursor), and move the remaining lines up. The bottom <i>n</i> lines are cleared. The current cursor position remains the same.</p> <p>With the <code>insertln()</code> and <code>winsertln()</code> routines, a blank line is inserted above the current line and the bottom line is lost. This does not imply use of a hardware insert line feature.</p> | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . Note that all but <code>winsdelln()</code> may be macros. | | | | |

NAME insdelln, winsdelln – insert/delete lines to/from the window

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

int insdelln(int n);

int winsdelln(WINDOW *win, int n);
```

PARAMETERS

n Is the number of lines to insert or delete (positive *n* inserts; negative *n* deletes).

win Is a pointer to the window in which to insert or delete a line.

DESCRIPTION The insdelln() and winsdelln() functions insert or delete blank lines in stdscr or *win*, respectively. When *n* is positive, *n* lines are added before the current line and the bottom *n* lines are lost; when *n* is negative, *n* lines are deleted starting with the current line, the remaining lines are moved up, and the bottom *n* lines are cleared. The position of the cursor does not change.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO deleteln(3XCURSES), insertln(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

insertln(3CURSES)

| NAME | <code> curs_deleteln, deleteln, wdeleteln, insdelln, winsdelln, insertln, winsertln</code> – delete and insert lines in a curses window | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int deleteln(void); int wdeleteln(WINDOW *win); int insdelln(int <i>n</i>); int winsdelln(WINDOW *win, int <i>n</i>); int insertln(void); int winsertln(WINDOW *win);</pre> | | | | |
| DESCRIPTION | <p>With the <code>deleteln()</code> and <code>wdeleteln()</code> routines, the line under the cursor in the window is deleted; all lines below the current line are moved up one line. The bottom line of the window is cleared. The cursor position does not change. This does not imply use of a hardware delete line feature.</p> <p>With the <code>insdelln()</code> and <code>winsdelln()</code> routines, for positive <i>n</i>, insert <i>n</i> lines into the specified window above the current line. The <i>n</i> bottom lines are lost. For negative <i>n</i>, delete <i>n</i> lines (starting with the one under the cursor), and move the remaining lines up. The bottom <i>n</i> lines are cleared. The current cursor position remains the same.</p> <p>With the <code>insertln()</code> and <code>winsertln()</code> routines, a blank line is inserted above the current line and the bottom line is lost. This does not imply use of a hardware insert line feature.</p> | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . Note that all but <code>winsdelln()</code> may be macros. | | | | |

NAME insertln, winsertln – insert a line in a window

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

int insertln(void);

int winsertln(WINDOW *win);
```

PARAMETERS *win* Is a pointer to the window in which to insert the line.

DESCRIPTION The insertln() and winsertln() functions insert a blank line before the current line in stdscr or *win*, respectively. The new line becomes the current line. The current line and all lines after it in the window are moved down one line. The bottom line in the window is discarded.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO bkgdset(3XCURSES), deleteln(3XCURSES), insdelln(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

insnstr(3CURSES)

| NAME | <code> curs_insstr, insstr, insnstr, winsstr, winsnstr, mvinsstr, mvinsnstr, mvwinsstr, mvwinsnstr </code> – insert string before character under the cursor in a curses window | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int insstr(char *str); int insnstr(char *str, int n); int winsstr(WINDOW *win, char *str); int winsnstr(WINDOW *win, char *str, int n); int mvinsstr(int y, int x, char *str); int mvinsnstr(int y, int x, char *str, int n); int mvwinsstr(WINDOW *win, int y, int x, char *str); int mvwinsnstr(WINDOW *win, int y, int x, char *str, int n);</pre> | | | | |
| DESCRIPTION | <p>With these routines, a character string (as many characters as will fit on the line) is inserted before the character under the cursor. All characters to the right of the cursor are moved to the right, with the possibility of the rightmost characters on the line being lost. The cursor position does not change (after moving to <i>y, x</i>, if specified). (This does not imply use of the hardware insert character feature.) The four routines with <i>n</i> as the last argument insert at most <i>n</i> characters. If <i>n</i> ≤ 0, then the entire string is inserted.</p> <p>If a character in <i>str</i> is a tab, newline, carriage return or backspace, the cursor is moved appropriately within the window. A newline also does a <code>clrtoeol()</code> before moving. Tabs are considered to be at every eighth column. If a character in <i>str</i> is another control character, it is drawn in the <code>^X</code> notation. Calling <code>winch()</code> after adding a control character (and moving to it, if necessary) does not return the control character, but instead returns the representation of the control character.</p> | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code> curs_clear(3CURSES), curs_inch(3CURSES), curses(3CURSES), attributes(5) </code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . | | | | |

`insnstr(3COURSE)`

Note that all but `winsnstr()` may be macros.

insnstr(3XCURSES)

| | | | | | | | | | | | |
|--------------------|---|------------|--|----------|--|----------|---|----------|--|------------|---|
| NAME | <code>insnstr</code> , <code>insstr</code> , <code>mvinsnstr</code> , <code>mvinsstr</code> , <code>mvwinsnstr</code> , <code>mvwinsstr</code> , <code>winsnstr</code> , <code>winsstr</code> – insert a multibyte character string | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int insnstr(const char *str, int n); int insstr(const char *str); int mvinsnstr(int y, int x, const char *str, int n); int mvinsstr(int y, int x, const char *str); int mvwinsnstr(WINDOW *win, int y, int x, const char *str, int n); int mvwinsstr(WINDOW *win, int y, int x, const char *str); int winsnstr(WINDOW *win, const char *str, int n); int winsstr(WINDOW *win, const char *str);</pre> | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>str</i></td><td>Is a pointer to the string to be inserted.</td></tr><tr><td><i>n</i></td><td>Is the number of characters not to exceed when inserting <i>str</i>. If <i>n</i> is less than 1, the entire string is inserted.</td></tr><tr><td><i>y</i></td><td>Is the y (row) coordinate of the starting position of the string.</td></tr><tr><td><i>x</i></td><td>Is the x (column) coordinate of the starting position of the string.</td></tr><tr><td><i>win</i></td><td>Is a pointer to the window in which the string is to be inserted.</td></tr></table> | <i>str</i> | Is a pointer to the string to be inserted. | <i>n</i> | Is the number of characters not to exceed when inserting <i>str</i> . If <i>n</i> is less than 1, the entire string is inserted. | <i>y</i> | Is the y (row) coordinate of the starting position of the string. | <i>x</i> | Is the x (column) coordinate of the starting position of the string. | <i>win</i> | Is a pointer to the window in which the string is to be inserted. |
| <i>str</i> | Is a pointer to the string to be inserted. | | | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when inserting <i>str</i> . If <i>n</i> is less than 1, the entire string is inserted. | | | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of the starting position of the string. | | | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of the starting position of the string. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the string is to be inserted. | | | | | | | | | | |
| DESCRIPTION | <p>The <code>insstr()</code> function inserts <i>str</i> at the current cursor position of the <code>stdscr</code> window. The <code>winsstr()</code> function performs the identical action, but in window <i>win</i>. The <code>mvinsstr()</code> and <code>mvwinsstr()</code> functions insert the character string at the starting position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <i>win</i>).</p> <p>The <code>insnstr()</code>, <code>winsnstr()</code>, <code>mvinsnstr()</code>, and <code>mvwinsnstr()</code> functions insert <i>n</i> characters to the window or as many as will fit on the line. If <i>n</i> is less than 1, the entire string is inserted or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <i>x</i> and <i>y</i> parameters.</p> <p>All characters to the right of inserted characters are moved to the right. Characters that don't fit on the current line are discarded. The cursor is left at the point of insertion.</p> | | | | | | | | | | |

insnstr(3XCURSES)

If a character in *str* is a newline, carriage return, backspace, or tab, the cursor is moved appropriately. The cursor is moved to the next tab stop for each tab character (by default, tabs are eight characters apart). If the character is a control character other than those previously mentioned, the character is inserted using \hat{x} notation, where x is a printable character. `clrtoeol(3XCURSES)` is automatically done before a newline.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addchstr(3XCURSES)`, `addstr(3XCURSES)`, `clrtoeol(3XCURSES)`, `ins_nwstr(3XCURSES)`, `insch(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

inswstr(3CURSES)

| NAME | <p> <code> curs_inswstr, inswstr, insnwstr, winswstr, winsnwstr, mvinswstr, mvinsnwstr, mvwinswstr, mvwinsnwstr</code> – insert <code>wchar_t</code> string before character under the cursor in a curses window </p> | | | | |
|---------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre> cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int inswstr(wchar_t *wstr); int insnwstr(wchar_t *wstr, int n); int winswstr(WINDOW *win, wchar_t *wstr); int winsnwstr(WINDOW *win, wchar_t *wstr, int n); int mvinswstr(int y, int x, wchar_t *wstr); int mvinsnwstr(int y, int x, wchar_t *wstr, int n); int mvwinswstr(WINDOW *win, int y, int x, wchar_t *wstr); int mvwinsnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n); </pre> | | | | |
| DESCRIPTION | <p> These routines insert a <code>wchar_t</code> character string (as many <code>wchar_t</code> characters as will fit on the line) before the character under the cursor. All characters to the right of the cursor are moved to the right, with the possibility of the rightmost characters on the line being lost. The cursor position does not change (after moving to <code>y, x</code>, if specified). (This does not imply use of the hardware insert character feature.) The four routines with <code>n</code> as the last argument insert at most <code>n</code> <code>wchar_t</code> characters. If <code>n <= 0</code>, then the entire string is inserted. </p> <p> If a character in <code>wstr</code> is a tab, newline, carriage return, or backspace, the cursor is moved appropriately within the window. A newline also does a <code>clrtoeol(3CURSES)</code> before moving. Tabs are considered to be at every eighth column. If a character in <code>wstr</code> is another control character, it is drawn in the <code>^X</code> notation. Calling <code>winwch(3CURSES)</code> after adding a control character (and moving to it, if necessary) does not return the control character, but instead returns the representation of the control character. </p> | | | | |
| RETURN VALUE | <p> All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. </p> | | | | |
| ATTRIBUTES | <p> See <code>attributes(5)</code> for descriptions of the following attributes: </p> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr> <th style="text-align: center; padding: 5px;">ATTRIBUTE TYPE</th> <th style="text-align: center; padding: 5px;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td style="padding: 5px;">MT-Level</td> <td style="padding: 5px;">Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <p> <code>clrtoeol(3CURSES)</code>, <code>curses(3CURSES)</code>, <code>winwch(3CURSES)</code>, <code>attributes(5)</code> </p> | | | | |
| NOTES | <p> The header file <code><curses.h></code> automatically includes the header files <code><stdio.h></code>, <code><unctrl.h></code> and <code><wchar.h></code>. </p> | | | | |

`insnwstr(3COURSE)`

Note that all but `winsnwstr()` may be macros.

ins_nwstr(3XCURSES)

| | | | | | | | | | | | |
|--------------------|---|-------------|--|----------|---|----------|---|----------|--|------------|---|
| NAME | ins_nwstr, ins_wstr, mvins_nwstr, mvins_wstr, mvwins_nwstr, mvwins_wstr, wins_nwstr, wins_wstr – insert a wide character string | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int ins_nwstr(const wchar_t *wstr, int n); int ins_wstr(const wchar_t *wstr); int mvins_nwstr(int y, int x, const wchar_t *wstr, int n); int mvins_wstr(int y, int x, const wchar_t *wstr); int mvwins_nwstr(WINDOW *win, int y, int x, const wchar_t *wstr, int n); int mvwins_wstr(WINDOW *win, int y, int x, const wchar_t *wstr); int wins_nwstr(WINDOW *win, const wchar_t *wstr, int n); int wins_wstr(WINDOW *win, const wchar_t *wstr);</pre> | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>wstr</i></td><td>Is a pointer to the string to be inserted.</td></tr><tr><td><i>n</i></td><td>Is the number of characters not to exceed when inserting <i>wstr</i>. If <i>n</i> is less than 1, the entire string is inserted.</td></tr><tr><td><i>y</i></td><td>Is the y (row) coordinate of the starting position of the string.</td></tr><tr><td><i>x</i></td><td>Is the x (column) coordinate of the starting position of the string.</td></tr><tr><td><i>win</i></td><td>Is a pointer to the window in which the string is to be inserted.</td></tr></table> | <i>wstr</i> | Is a pointer to the string to be inserted. | <i>n</i> | Is the number of characters not to exceed when inserting <i>wstr</i> . If <i>n</i> is less than 1, the entire string is inserted. | <i>y</i> | Is the y (row) coordinate of the starting position of the string. | <i>x</i> | Is the x (column) coordinate of the starting position of the string. | <i>win</i> | Is a pointer to the window in which the string is to be inserted. |
| <i>wstr</i> | Is a pointer to the string to be inserted. | | | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when inserting <i>wstr</i> . If <i>n</i> is less than 1, the entire string is inserted. | | | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of the starting position of the string. | | | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of the starting position of the string. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the string is to be inserted. | | | | | | | | | | |
| DESCRIPTION | <p>The <code>ins_wstr()</code> function inserts <i>wstr</i> at the current cursor position of the <code>stdscr</code> window. The <code>wins_wstr()</code> function performs the identical action, but in window <i>win</i>. The <code>mvins_wstr()</code> and <code>mvwins_wstr()</code> functions insert <i>wstr</i> string at the starting position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>ins_nwstr()</code>, <code>wins_nwstr()</code>, <code>mvins_nwstr()</code>, and <code>mvwins_nwstr()</code> functions insert <i>n</i> characters to the window or as many as will fit on the line. If <i>n</i> is less than 1, the entire string is inserted or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <i>x</i> and <i>y</i> parameters.</p> | | | | | | | | | | |

ins_nwstr(3XCURSES)

All characters to the right of inserted characters are moved to the right. Characters that don't fit on the current line are discarded. The cursor is left at the point of insertion.

If a character in *wstr* is a newline, carriage return, backspace, or tab, the cursor is moved appropriately. The cursor is moved to the next tab stop for each tab character (by default, tabs are eight characters apart). If the character is a control character other than those previously mentioned, the character is inserted using \hat{x} notation, where x is a printable character. `clrtoeol(3XCURSES)` is automatically done before a newline.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `add_wchnstr(3XCURSES)`, `addnwstr(3XCURSES)`, `clrtoeol(3XCURSES)`, `ins_wch(3XCURSES)`, `insnstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

insstr(3CURSES)

| NAME | <code> curs_insstr, insstr, insnstr, winsstr, winsnstr, mvinsstr, mvinsnstr, mvwinsstr, mvwinsnstr </code> – insert string before character under the cursor in a curses window | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int insstr(char *str); int insnstr(char *str, int n); int winsstr(WINDOW *win, char *str); int winsnstr(WINDOW *win, char *str, int n); int mvinsstr(int y, int x, char *str); int mvinsnstr(int y, int x, char *str, int n); int mvwinsstr(WINDOW *win, int y, int x, char *str); int mvwinsnstr(WINDOW *win, int y, int x, char *str, int n);</pre> | | | | |
| DESCRIPTION | <p>With these routines, a character string (as many characters as will fit on the line) is inserted before the character under the cursor. All characters to the right of the cursor are moved to the right, with the possibility of the rightmost characters on the line being lost. The cursor position does not change (after moving to <i>y, x</i>, if specified). (This does not imply use of the hardware insert character feature.) The four routines with <i>n</i> as the last argument insert at most <i>n</i> characters. If <i>n</i> ≤ 0, then the entire string is inserted.</p> <p>If a character in <i>str</i> is a tab, newline, carriage return or backspace, the cursor is moved appropriately within the window. A newline also does a <code>clrtoeol()</code> before moving. Tabs are considered to be at every eighth column. If a character in <i>str</i> is another control character, it is drawn in the ^X notation. Calling <code>winch()</code> after adding a control character (and moving to it, if necessary) does not return the control character, but instead returns the representation of the control character.</p> | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code> curs_clear(3CURSES), curs_inch(3CURSES), curses(3CURSES), attributes(5) </code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . | | | | |

`instr(3CURSES)`

Note that all but `winsnstr()` may be macros.

insstr(3XCURSES)

| | | | | | | | | | | | |
|--------------------|--|------------|--|----------|--|----------|---|----------|--|------------|---|
| NAME | insnstr, insstr, mvinsnstr, mvinsstr, mvwinsnstr, mvwinsstr, winsnstr, winsstr – insert a multibyte character string | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int insnstr(const char *str, int n); int insstr(const char *str); int mvinsnstr(int y, int x, const char *str, int n); int mvinsstr(int y, int x, const char *str); int mvwinsnstr(WINDOW *win, int y, int x, const char *str, int n); int mvwinsstr(WINDOW *win, int y, int x, const char *str); int winsnstr(WINDOW *win, const char *str, int n); int winsstr(WINDOW *win, const char *str);</pre> | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>str</i></td><td>Is a pointer to the string to be inserted.</td></tr><tr><td><i>n</i></td><td>Is the number of characters not to exceed when inserting <i>str</i>. If <i>n</i> is less than 1, the entire string is inserted.</td></tr><tr><td><i>y</i></td><td>Is the y (row) coordinate of the starting position of the string.</td></tr><tr><td><i>x</i></td><td>Is the x (column) coordinate of the starting position of the string.</td></tr><tr><td><i>win</i></td><td>Is a pointer to the window in which the string is to be inserted.</td></tr></table> | <i>str</i> | Is a pointer to the string to be inserted. | <i>n</i> | Is the number of characters not to exceed when inserting <i>str</i> . If <i>n</i> is less than 1, the entire string is inserted. | <i>y</i> | Is the y (row) coordinate of the starting position of the string. | <i>x</i> | Is the x (column) coordinate of the starting position of the string. | <i>win</i> | Is a pointer to the window in which the string is to be inserted. |
| <i>str</i> | Is a pointer to the string to be inserted. | | | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when inserting <i>str</i> . If <i>n</i> is less than 1, the entire string is inserted. | | | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of the starting position of the string. | | | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of the starting position of the string. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the string is to be inserted. | | | | | | | | | | |
| DESCRIPTION | <p>The <code>insstr()</code> function inserts <i>str</i> at the current cursor position of the <code>stdscr</code> window. The <code>winsstr()</code> function performs the identical action, but in window <i>win</i>. The <code>mvinsnstr()</code> and <code>mvwinsstr()</code> functions insert the character string at the starting position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <i>win</i>).</p> <p>The <code>insnstr()</code>, <code>winsnstr()</code>, <code>mvinsnstr()</code>, and <code>mvwinsnstr()</code> functions insert <i>n</i> characters to the window or as many as will fit on the line. If <i>n</i> is less than 1, the entire string is inserted or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <i>x</i> and <i>y</i> parameters.</p> <p>All characters to the right of inserted characters are moved to the right. Characters that don't fit on the current line are discarded. The cursor is left at the point of insertion.</p> | | | | | | | | | | |

insstr(3XCURSES)

If a character in *str* is a newline, carriage return, backspace, or tab, the cursor is moved appropriately. The cursor is moved to the next tab stop for each tab character (by default, tabs are eight characters apart). If the character is a control character other than those previously mentioned, the character is inserted using \hat{x} notation, where *x* is a printable character. `clrtoeol(3XCURSES)` is automatically done before a newline.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addchstr(3XCURSES)`, `addstr(3XCURSES)`, `clrtoeol(3XCURSES)`, `ins_nwstr(3XCURSES)`, `insch(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

instr(3CURSES)

| NAME | curs_instr, instr, innstr, winstr, winnstr, mvinstr, mvinnstr, mvwinstr, mvwinnstr – get a string of characters from a curses window | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int instr(char *str); int innstr(char *str, int n); int winstr(WINDOW *win, char *str); int winnstr(WINDOW *win, char *str, int n); int mvinstr(int y, int x, char *str); int mvinnstr(int y, int x, char *str, int n); int mvwinstr(WINDOW *win, int y, int x, char *str); int mvwinnstr(WINDOW *win, int y, int x, char *str, int n);</pre> | | | | |
| DESCRIPTION | These routines return a string of characters in <i>str</i> , starting at the current cursor position in the named window and ending at the right margin of the window. Attributes are stripped from the characters. The four functions with <i>n</i> as the last argument return the string at most <i>n</i> characters long. | | | | |
| RETURN VALUES | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), attributes(5) | | | | |
| NOTES | The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>. <p>Note that all routines except winnstr() may be macros.</p> | | | | |

| | | | | | | | | | | | |
|--------------------|--|------------|---|----------|--|----------|--|----------|---|------------|--|
| NAME | innstr, instr, mvinnstr, mvinstr, mvwinnstr, mvwinstr, winnstr, winstr – retrieve a multibyte character string (without rendition) | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int innstr(char *str, int n); int instr(char *str); int mvinnstr(int y, int x, char *str, int n); int mvinstr(int y, int x, char *str); int mvwinnstr(WINDOW *win, int y, int x, char *str, int n); int mvwinstr(WINDOW *win, int y, int x, char *str); int winstr(WINDOW *win, char *str); int winnstr(WINDOW *win, char *str, int n);</pre> | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>str</i></td> <td>Is a pointer to an object that can hold the retrieved multibyte character string.</td> </tr> <tr> <td><i>n</i></td> <td>Is the number of characters not to exceed when retrieving <i>str</i>.</td> </tr> <tr> <td><i>y</i></td> <td>Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved.</td> </tr> <tr> <td><i>x</i></td> <td>Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved.</td> </tr> <tr> <td><i>win</i></td> <td>Is a pointer to the window in which the string is to be retrieved.</td> </tr> </table> | <i>str</i> | Is a pointer to an object that can hold the retrieved multibyte character string. | <i>n</i> | Is the number of characters not to exceed when retrieving <i>str</i> . | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | <i>win</i> | Is a pointer to the window in which the string is to be retrieved. |
| <i>str</i> | Is a pointer to an object that can hold the retrieved multibyte character string. | | | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when retrieving <i>str</i> . | | | | | | | | | | |
| <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | | | | | | | | | | |
| <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the string is to be retrieved. | | | | | | | | | | |
| DESCRIPTION | <p>The <code>instr()</code> and <code>winstr()</code> functions retrieve a multibyte character string (without attributes) starting at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively, and ending at the right margin. The <code>mvinstr()</code> and <code>mvwinstr()</code> functions retrieve a multibyte character string located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>innstr()</code>, <code>winnstr()</code>, <code>mvinnstr()</code>, and <code>mvwinnstr()</code> functions retrieve at most <i>n</i> characters from the window <code>stdscr</code> and <i>win</i>, respectively. The former two functions retrieve the string starting at the current cursor position; the latter two commands return the string, starting at the position specified by the <i>x</i> and <i>y</i> parameters.</p> | | | | | | | | | | |

instr(3XCURSES)

All these functions store the retrieved string in the object pointed to by *str*. They only store complete multibyte characters. If the area pointed to by *str* is not large enough to hold at least one character, these functions fail.

Only the character portion of the character/rendition pair is returned. To return the complete character/rendition pair, use `winchstr()`.

ERRORS OK Successful completion.

ERR An error occurred.

USAGE All functions except `winnstr()` may be macros.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `inch(3XCURSES)`, `inchstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_inswch, inswch, winswch, mvinswch, mvwinswch – insert a <code>wchar_t</code> character before the character under the cursor in a curses window | | | | |
|---------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int inswch(chtype wch); int winswch(WINDOW *win, chtype wch); int mvinswch(int y, int x, chtype wch); int mvwinswch(WINDOW *win, int y, int x, chtype wch);</pre> | | | | |
| DESCRIPTION | <p>These routines insert the character <i>wch</i>, holding a <code>wchar_t</code> character, before the character under the cursor. All characters to the right of the cursor are moved one space to the right, with the possibility of the rightmost character on the line being lost. The cursor position does not change (after moving to <i>y, x</i>, if specified). (This does not imply use of the hardware insert character feature.)</p> | | | | |
| RETURN VALUE | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">MT-Level</td> <td style="text-align: center;">Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header file <code><curses.h></code> automatically includes the header files <code><stdio.h></code>, <code><unctrl.h></code> and <code><widec.h></code>.</p> <p>Note that <code>inswch()</code>, <code>mvinswch()</code>, and <code>mvwinswch()</code> may be macros.</p> <p>None of these routines can use the color attribute in <code>chtype</code>.</p> | | | | |

ins_wch(3XCURSES)

NAME | ins_wch, wins_wch, mvins_wch, mvwins_wch – insert a complex character

SYNOPSIS | `cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \`
`-R /usr/xpg4/lib -lcurses [library...]`

c89 | `[flag...] file... -lcurses [library...]`

`#include <curses.h>`

`int ins_wch(const cchar_t *wch);`

`int mvins_wch(int y, int x, const cchar_t *wch);`

`int mvwins_wch(WINDOW *win, int y, int x, const cchar_t *wch);`

`int wins_wch(WINDOW *win, const cchar_t *wch);`

PARAMETERS | *wch* | Is the complex character to be inserted.

| *y* | Is the y (row) coordinate of the position of the character.

| *x* | Is the x (column) coordinate of the position of the character.

| *win* | Is a pointer to the window in which the character is to be inserted.

DESCRIPTION | The `ins_wch()` function inserts the complex character *wch* at the current cursor position of the `stdscr` window. The `wins_wch()` function performs the identical action but in window *win*. The `mvins_wch()` and `mvwins_wch()` functions insert the character at the position indicated by the *x* (column) and *y* (row) parameters (the former in the `stdscr` window; the latter in window *win*). The cursor position does not change.

| All characters to the right of the inserted character are moved right one character. The last character on the line is deleted.

| Insertions and deletions occur at the character level. The cursor is adjusted to the first column of the character prior to the the operation.

RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR.

ERRORS | None.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

ins_wch(3XCURSES)

SEE ALSO add_wch(3XCURSES), ins_nwstr(3XCURSES), libcurses(3XCURSES),
attributes(5), standards(5)

inswstr(3CURSES)

| NAME | <code> curs_inswstr, inswstr, insnwstr, winswstr, winsnwstr, mvinswstr, mvinsnwstr, mvwinswstr, mvwinsnwstr </code> – insert <code>wchar_t</code> string before character under the cursor in a curses window | | | | |
|---------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int inswstr(wchar_t *wstr); int insnwstr(wchar_t *wstr, int n); int winswstr(WINDOW *win, wchar_t *wstr); int winsnwstr(WINDOW *win, wchar_t *wstr, int n); int mvinswstr(int y, int x, wchar_t *wstr); int mvinsnwstr(int y, int x, wchar_t *wstr, int n); int mvwinswstr(WINDOW *win, int y, int x, wchar_t *wstr); int mvwinsnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);</pre> | | | | |
| DESCRIPTION | <p>These routines insert a <code>wchar_t</code> character string (as many <code>wchar_t</code> characters as will fit on the line) before the character under the cursor. All characters to the right of the cursor are moved to the right, with the possibility of the rightmost characters on the line being lost. The cursor position does not change (after moving to <code>y, x</code>, if specified). (This does not imply use of the hardware insert character feature.) The four routines with <code>n</code> as the last argument insert at most <code>n</code> <code>wchar_t</code> characters. If <code>n</code> ≤ 0, then the entire string is inserted.</p> <p>If a character in <code>wstr</code> is a tab, newline, carriage return, or backspace, the cursor is moved appropriately within the window. A newline also does a <code>clrtoeol(3CURSES)</code> before moving. Tabs are considered to be at every eighth column. If a character in <code>wstr</code> is another control character, it is drawn in the <code>^X</code> notation. Calling <code>winwch(3CURSES)</code> after adding a control character (and moving to it, if necessary) does not return the control character, but instead returns the representation of the control character.</p> | | | | |
| RETURN VALUE | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>clrtoeol(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>winwch(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header file <code><curses.h></code> automatically includes the header files <code><stdio.h></code> , <code><unctrl.h></code> and <code><widec.h></code> . | | | | |

`inswstr(3CURSES)`

Note that all but `winswstr()` may be macros.

ins_wstr(3XCURSES)

| | | | | | | | | | | | |
|--------------------|---|-------------|--|----------|---|----------|---|----------|--|------------|---|
| NAME | ins_nwstr, ins_wstr, mvins_nwstr, mvins_wstr, mvwins_nwstr, mvwins_wstr, wins_nwstr, wins_wstr – insert a wide character string | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int ins_nwstr(const wchar_t *wstr, int n); int ins_wstr(const wchar_t *wstr); int mvins_nwstr(int y, int x, const wchar_t *wstr, int n); int mvins_wstr(int y, int x, const wchar_t *wstr); int mvwins_nwstr(WINDOW *win, int y, int x, const wchar_t *wstr, int n); int mvwins_wstr(WINDOW *win, int y, int x, const wchar_t *wstr); int wins_nwstr(WINDOW *win, const wchar_t *wstr, int n); int wins_wstr(WINDOW *win, const wchar_t *wstr);</pre> | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>wstr</i></td><td>Is a pointer to the string to be inserted.</td></tr><tr><td><i>n</i></td><td>Is the number of characters not to exceed when inserting <i>wstr</i>. If <i>n</i> is less than 1, the entire string is inserted.</td></tr><tr><td><i>y</i></td><td>Is the y (row) coordinate of the starting position of the string.</td></tr><tr><td><i>x</i></td><td>Is the x (column) coordinate of the starting position of the string.</td></tr><tr><td><i>win</i></td><td>Is a pointer to the window in which the string is to be inserted.</td></tr></table> | <i>wstr</i> | Is a pointer to the string to be inserted. | <i>n</i> | Is the number of characters not to exceed when inserting <i>wstr</i> . If <i>n</i> is less than 1, the entire string is inserted. | <i>y</i> | Is the y (row) coordinate of the starting position of the string. | <i>x</i> | Is the x (column) coordinate of the starting position of the string. | <i>win</i> | Is a pointer to the window in which the string is to be inserted. |
| <i>wstr</i> | Is a pointer to the string to be inserted. | | | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when inserting <i>wstr</i> . If <i>n</i> is less than 1, the entire string is inserted. | | | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of the starting position of the string. | | | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of the starting position of the string. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the string is to be inserted. | | | | | | | | | | |
| DESCRIPTION | <p>The <code>ins_wstr()</code> function inserts <i>wstr</i> at the current cursor position of the <code>stdscr</code> window. The <code>wins_wstr()</code> function performs the identical action, but in window <i>win</i>. The <code>mvins_wstr()</code> and <code>mvwins_wstr()</code> functions insert <i>wstr</i> string at the starting position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>ins_nwstr()</code>, <code>wins_nwstr()</code>, <code>mvins_nwstr()</code>, and <code>mvwins_nwstr()</code> functions insert <i>n</i> characters to the window or as many as will fit on the line. If <i>n</i> is less than 1, the entire string is inserted or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <i>x</i> and <i>y</i> parameters.</p> | | | | | | | | | | |

ins_wstr(3XCURSES)

All characters to the right of inserted characters are moved to the right. Characters that don't fit on the current line are discarded. The cursor is left at the point of insertion.

If a character in *wstr* is a newline, carriage return, backspace, or tab, the cursor is moved appropriately. The cursor is moved to the next tab stop for each tab character (by default, tabs are eight characters apart). If the character is a control character other than those previously mentioned, the character is inserted using \hat{x} notation, where x is a printable character. `clrtoeol(3XCURSES)` is automatically done before a newline.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `add_wchnstr(3XCURSES)`, `addnwstr(3XCURSES)`, `clrtoeol(3XCURSES)`, `ins_wch(3XCURSES)`, `insnstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

intrflush(3CURSES)

| | |
|--------------------|---|
| NAME | <code>curs_inopts</code> , <code>cbreak</code> , <code>nocbreak</code> , <code>echo</code> , <code>noecho</code> , <code>halfdelay</code> , <code>intrflush</code> , <code>keypad</code> , <code>meta</code> , <code>nodelay</code> , <code>notimeout</code> , <code>raw</code> , <code>noraw</code> , <code>noqiflush</code> , <code>qiflush</code> , <code>timeout</code> , <code>wtimeout</code> , <code>typeahead</code> – curses terminal input option control routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int cbreak(void); int nocbreak(void); int echo(void); int noecho(void); int halfdelay(int <i>tenths</i>); int intrflush(WINDOW *<i>win</i>, bool <i>bf</i>); int keypad(WINDOW *<i>win</i>, bool <i>bf</i>); int meta(WINDOW *<i>win</i>, bool <i>bf</i>); int nodelay(WINDOW *<i>win</i>, bool <i>bf</i>); int notimeout(WINDOW *<i>win</i>, bool <i>bf</i>); int raw(void); int noraw(void); void noqiflush(void); void qiflush(void); void timeout(int <i>delay</i>); void wtimeout(WINDOW *<i>win</i>, int <i>delay</i>); int typeahead(int <i>fildes</i>);</pre> |
| DESCRIPTION | <p>The <code>cbreak()</code> and <code>nocbreak()</code> routines put the terminal into and out of <code>cbreak()</code> mode, respectively. In this mode, characters typed by the user are immediately available to the program, and erase/kill character-processing is not performed. When out of this mode, the tty driver buffers the typed characters until a newline or carriage return is typed. Interrupt and flow control characters are unaffected by this mode. Initially the terminal may or may not be in <code>cbreak()</code> mode, as the mode is inherited; therefore, a program should call <code>cbreak()</code> or <code>nocbreak()</code> explicitly. Most interactive programs using curses set the <code>cbreak()</code> mode.</p> <p>Note that <code>cbreak()</code> overrides <code>raw()</code>. (See <code>curs_getch(3CURSES)</code> for a discussion of how these routines interact with <code>echo()</code> and <code>noecho()</code>.)</p> <p>The <code>echo()</code> and <code>noecho()</code> routines control whether characters typed by the user are echoed by <code>getch()</code> as they are typed. Echoing by the tty driver is always disabled, but initially <code>getch()</code> is in echo mode, so characters typed are echoed. Authors of</p> |

most interactive programs prefer to do their own echoing in a controlled area of the screen, or not to echo at all, so they disable echoing by calling `noecho()`. (See `curs_getch(3CURSES)` for a discussion of how these routines interact with `cbreak()` and `nocbreak()`.)

The `halfdelay()` routine is used for half-delay mode, which is similar to `cbreak()` mode in that characters typed by the user are immediately available to the program. However, after blocking for *tenths* tenths of seconds, `ERR` is returned if nothing has been typed. The value of *tenths* must be a number between 1 and 255. Use `nocbreak()` to leave half-delay mode.

If the `intrflush()` option is enabled, (*bf* is `TRUE`), when an interrupt key is pressed on the keyboard (interrupt, break, quit) all output in the tty driver queue will be flushed, giving the effect of faster response to the interrupt, but causing `curses` to have the wrong idea of what is on the screen. Disabling (*bf* is `FALSE`), the option prevents the flush. The default for the option is inherited from the tty driver settings. The window argument is ignored.

The `keypad()` option enables the keypad of the user's terminal. If enabled (*bf* is `TRUE`), the user can press a function key (such as an arrow key) and `wgetch()` returns a single value representing the function key, as in `KEY_LEFT`. If disabled (*bf* is `FALSE`), `curses` does not treat function keys specially and the program has to interpret the escape sequences itself. If the keypad in the terminal can be turned on (made to transmit) and off (made to work locally), turning on this option causes the terminal keypad to be turned on when `wgetch()` is called. The default value for keypad is false.

Initially, whether the terminal returns 7 or 8 significant bits on input depends on the control mode of the tty driver (see `termio(7I)`). To force 8 bits to be returned, invoke `meta(win, TRUE)`. To force 7 bits to be returned, invoke `meta(win, FALSE)`. The window argument, *win*, is always ignored. If the terminfo capabilities `smm` (`meta_on`) and `rmm` (`meta_off`) are defined for the terminal, `smm` is sent to the terminal when `meta(win, TRUE)` is called and `rmm` is sent when `meta(win, FALSE)` is called.

The `nodelay()` option causes `getch()` to be a non-blocking call. If no input is ready, `getch()` returns `ERR`. If disabled (*bf* is `FALSE`), `getch()` waits until a key is pressed.

While interpreting an input escape sequence, `wgetch()` sets a timer while waiting for the next character. If `notimeout(win, TRUE)` is called, then `wgetch()` does not set a timer. The purpose of the timeout is to differentiate between sequences received from a function key and those typed by a user.

With the `raw()` and `noraw()` routines, the terminal is placed into or out of raw mode. Raw mode is similar to `cbreak()` mode, in that characters typed are immediately passed through to the user program. The differences are that in raw mode, the interrupt, quit, suspend, and flow control characters are all passed through uninterpreted, instead of generating a signal. The behavior of the `BREAK` key depends on other bits in the tty driver that are not set by `curses`.

intrflush(3CURSES)

When the `noqiflush()` routine is used, normal flush of input and output queues associated with the `INTR`, `QUIT` and `SUSP` characters will not be done (see `termio(7I)`). When `qiflush()` is called, the queues will be flushed when these control characters are read.

The `timeout()` and `wtimeout()` routines set blocking or non-blocking read for a given window. If *delay* is negative, blocking read is used (that is, waits indefinitely for input). If *delay* is zero, then non-blocking read is used (that is, read returns `ERR` if no input is waiting). If *delay* is positive, then read blocks for *delay* milliseconds, and returns `ERR` if there is still no input. Hence, these routines provide the same functionality as `nodelay()`, plus the additional capability of being able to block for only *delay* milliseconds (where *delay* is positive).

`curses` does "line-breakout optimization" by looking for typeahead periodically while updating the screen. If input is found, and it is coming from a tty, the current update is postponed until `refresh()` or `doupdate()` is called again. This allows faster response to commands typed in advance. Normally, the input FILE pointer passed to `newterm()`, or `stdin` in the case that `initscr()` was used, will be used to do this typeahead checking. The `typeahead()` routine specifies that the file descriptor *fildev* is to be used to check for typeahead instead. If *fildev* is `-1`, then no typeahead checking is done.

RETURN VALUES

All routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_getch(3CURSES)`, `curs_initscr(3CURSES)`, `curses(3CURSES)`, `attributes(5)`, `termio(7I)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `echo()`, `noecho()`, `halfdelay()`, `intrflush()`, `meta()`, `nodelay()`, `notimeout()`, `noqiflush()`, `qiflush()`, `timeout()`, and `wtimeout()` may be macros.

| NAME | intrflush – enable or disable flush on interrupt | | | | | | |
|----------------------|--|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int intrflush(WINDOW *win, bool bf);</pre> | | | | | | |
| PARAMETERS | <p><i>win</i> Is ignored.</p> <p><i>bf</i> Is a Boolean expression.</p> | | | | | | |
| DESCRIPTION | <p>The <code>intrflush()</code> function specifies whether pressing an interrupt key (interrupt, suspend, or quit) will flush the input buffer associated with the current screen. If the value of <i>bf</i> is <code>TRUE</code>, then flushing of the output buffer associated with the current screen will occur when an interrupt key (interrupt, suspend, or quit) is pressed. If the value of <i>bf</i> is <code>FALSE</code>, then no flushing of the buffer will occur when an interrupt key is pressed. The default for the option is inherited from the display driver settings. The <i>win</i> argument is ignored.</p> | | | | | | |
| RETURN VALUES | Upon successful completion, <code>intrflush()</code> returns <code>OK</code> . Otherwise, it returns <code>ERR</code> . | | | | | | |
| ERRORS | No errors are defined. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>flushinp(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>qiflush(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

inwch(3CURSES)

| NAME | <code> curs_inwch, inwch, winwch, mvwinch, mvwinwch </code> – get a <code>wchar_t</code> character and its attributes from a curses window | | | | |
|--------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> ctype inwch(void); ctype winwch(WINDOW *win); ctype mvwinch(int y, int x); ctype mvwinwch(WINDOW *win, int y, int x);</pre> | | | | |
| DESCRIPTION | These routines return the <code>wchar_t</code> character, of type <code>ctype</code> , at the current position in the named window. If any attributes are set for that position, their values are OR-ed into the value returned. Constants defined in <code><curses.h></code> can be used with the logical AND (<code>&</code>) operator to extract the character or attributes alone. | | | | |
| Attributes | The following bit-masks may be AND-ed with characters returned by <code>winwch()</code> . <code>A_WCHARTEXT</code> Bit-mask to extract character <code>A_WATTRIBUTES</code> Bit-mask to extract attributes | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header file <code><curses.h></code> automatically includes the header files <code><stdio.h></code> , <code><unctrl.h></code> and <code><widec.h></code> . Note that all of these routines may be macros. None of these routines can use the color attribute in <code>ctype</code> . | | | | |

| NAME | in_wch, mvin_wch, mvwin_wch, win_wch – retrieve a complex character (with rendition) | | | | | | | | |
|----------------------|---|----------------|---|---------------------|--|----------|---|------------|--|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int in_wch(cchar_t *wcv); int mvin_wch(int y, int x, cchar_t *wcv); int mvwin_wch(WINDOW *win, int y, cchar_t *wcv); int win_wch(WINDOW *win, cchar_t *wcv);</pre> | | | | | | | | |
| DESCRIPTION | <p>The in_wch() and win_wch() functions retrieve the complex character and its rendition located at the current cursor position of the stdscr window and window win, respectively. The mvin_wch() and mvwin_wch() functions retrieve the complex character and its rendition located at the position indicated by the x (column) and y (row) parameters (the former in the stdscr window; the latter in window win).</p> <p>All these functions store the retrieved character and its rendition in the object pointed to by wcv.</p> | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="vertical-align: top;"><i>wcv</i></td> <td>Is a pointer to an object that can store a complex character and its rendition.</td> </tr> <tr> <td style="vertical-align: top;"><i>y</i></td> <td>Is the y (row) coordinate of the position of the character to be returned.</td> </tr> <tr> <td style="vertical-align: top;"><i>x</i></td> <td>Is the x (column) coordinate of the position of the character to be returned.</td> </tr> <tr> <td style="vertical-align: top;"><i>win</i></td> <td>Is a pointer to the window that contains the character to be returned.</td> </tr> </table> | <i>wcv</i> | Is a pointer to an object that can store a complex character and its rendition. | <i>y</i> | Is the y (row) coordinate of the position of the character to be returned. | <i>x</i> | Is the x (column) coordinate of the position of the character to be returned. | <i>win</i> | Is a pointer to the window that contains the character to be returned. |
| <i>wcv</i> | Is a pointer to an object that can store a complex character and its rendition. | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of the position of the character to be returned. | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of the position of the character to be returned. | | | | | | | | |
| <i>win</i> | Is a pointer to the window that contains the character to be returned. | | | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | | | |
| ERRORS | None. | | | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | | | |
| Interface Stability | Standard | | | | | | | | |
| MT-Level | Unsafe | | | | | | | | |
| SEE ALSO | add_wch(3XCURSES), inch(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5) | | | | | | | | |

inwchnstr(3CURSES)

NAME | curs_inwchstr, inwchstr, inwchnstr, winwchstr, winwchnstr, mvinwchstr, mvinwchnstr, mvwinwchstr, mvwinwchnstr – get a string of wchar_t characters (and attributes) from a curses window

SYNOPSIS | `cc [flag ...] file ... -lcurses [library ..]`
`#include <curses.h>`

```
int inwchstr ( chtype *wchstr );
int inwchnstr ( chtype *wchstr, int n );
int winwchstr ( WINDOW *win, chtype *wchstr );
int winwchnstr ( WINDOW *win, chtype *wchstr, int n );
int mvinwchstr ( int y, int x, chtype *wchstr );
int mvinwchnstr ( int y, int x, chtype *wchstr, int n );
int mvwinwchstr ( WINDOW *win, int y, int x, chtype *wchstr );
int mvwinwchnstr ( WINDOW *win, int y, int x, chtype *wchstr, int n );
```

DESCRIPTION | These routines return a string of type chtype, holding wchar_t characters, starting at the current cursor position in the named window and ending at the right margin of the window. The four functions with *n* as the last argument, return the string at most *n* wchar_t characters long. Constants defined in <curses.h> can be used with the logical AND (&) operator to extract the wchar_t character or the attribute alone from any position in the wchstr (see curs_inwch(3CURSES)).

RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES | See attributes(5) for a description of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | curses(3CURSES), curs_inwch(3CURSES), attributes(5)

NOTES | The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <wdec.h>.

Note that all routines except winwchnstr() may be macros.

None of these routines can use the color attribute in chtype.

| | | | | | | | | | | | |
|--------------------|---|---------------|---|----------|---|----------|--|----------|---|------------|--|
| NAME | in_wchnstr, in_wchstr, mvin_wchnstr, mvin_wchstr, mvwin_wchnstr, mvwin_wchstr, win_wchnstr, win_wchstr – retrieve complex character string (with rendition) | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int in_wchnstr(cchar_t *wchstr, int n); int in_wchstr(cchar_t *wchstr); int mvin_wchnstr(int y, int x, cchar_t *wchstr, int n); int mvin_wchstr(int y, int x, cchar_t *wchstr); int mvwin_wchnstr(WINDOW *win, int y, int x, cchar_t *wchstr, int n); int mvwin_wchstr(WINDOW *win, int y, int x, cchar_t *wchstr); int win_wchnstr(WINDOW *win, cchar_t *wchstr, int n); int win_wchstr(WINDOW *win, cchar_t *wchstr);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The <code>in_wchstr()</code> and <code>win_wchstr()</code> functions retrieve a complex character string (with rendition) starting at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively, and ending at the right margin. The <code>mvin_wchstr()</code> and <code>mvwin_wchstr()</code> functions retrieve a complex character string located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>in_wchnstr()</code>, <code>win_wchnstr()</code>, <code>mvin_wchnstr()</code>, and <code>mvwin_wchnstr()</code> functions retrieve at most <i>n</i> characters from the window <code>stdscr</code> and <i>win</i>, respectively. The former two functions retrieve the string, starting at the current cursor position; the latter two commands retrieve the string, starting at the position specified by the <i>x</i> and <i>y</i> parameters.</p> <p>The retrieved character string (with renditions) is stored in the object pointed to by <i>wcval</i>.</p> | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>wchstr</i></td> <td>Is a pointer to an object where the retrieved complex character string can be stored.</td> </tr> <tr> <td><i>n</i></td> <td>Is the number of characters not to exceed when retrieving <i>wchstr</i>.</td> </tr> <tr> <td><i>y</i></td> <td>Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved.</td> </tr> <tr> <td><i>x</i></td> <td>Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved.</td> </tr> <tr> <td><i>win</i></td> <td>Is a pointer to the window in which the string is to be retrieved.</td> </tr> </table> | <i>wchstr</i> | Is a pointer to an object where the retrieved complex character string can be stored. | <i>n</i> | Is the number of characters not to exceed when retrieving <i>wchstr</i> . | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | <i>win</i> | Is a pointer to the window in which the string is to be retrieved. |
| <i>wchstr</i> | Is a pointer to an object where the retrieved complex character string can be stored. | | | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when retrieving <i>wchstr</i> . | | | | | | | | | | |
| <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | | | | | | | | | | |
| <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the string is to be retrieved. | | | | | | | | | | |

in_wchnstr(3XCURSES)

RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR.

ERRORS | None.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | `in_wch(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_inwchstr, inwchstr, inwchnstr, winwchstr, winwchnstr, mvinwchstr, mvinwchnstr, mvwinwchstr, mvwinwchnstr – get a string of wchar_t characters (and attributes) from a curses window | | | | |
|---------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int inwchstr(chtype *wchstr); int inwchnstr(chtype *wchstr, int n); int winwchstr(WINDOW *win, chtype *wchstr); int winwchnstr(WINDOW *win, chtype *wchstr, int n); int mvinwchstr(int y, int x, chtype *wchstr); int mvinwchnstr(int y, int x, chtype *wchstr, int n); int mvwinwchstr(WINDOW *win, int y, int x, chtype *wchstr); int mvwinwchnstr(WINDOW *win, int y, int x, chtype *wchstr, int n);</pre> | | | | |
| DESCRIPTION | These routines return a string of type chtype, holding wchar_t characters, starting at the current cursor position in the named window and ending at the right margin of the window. The four functions with <i>n</i> as the last argument, return the string at most <i>n</i> wchar_t characters long. Constants defined in <curses.h> can be used with the logical AND (&) operator to extract the wchar_t character or the attribute alone from any position in the <i>wchstr</i> (see curs_inwch(3CURSES)). | | | | |
| RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See attributes(5) for a description of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), curs_inwch(3CURSES), attributes(5) | | | | |
| NOTES | <p>The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <widec.h>.</p> <p>Note that all routines except winwchnstr() may be macros.</p> <p>None of these routines can use the color attribute in chtype.</p> | | | | |

in_wchstr(3XCURSES)

| | | | | | | | | | | | |
|---------------------|---|---------------------|---|----------------|---|----------------|--|----------------|---|------------------|--|
| NAME | in_wchnstr, in_wchstr, mvin_wchnstr, mvin_wchstr, mvwin_wchnstr, mvwin_wchstr, win_wchnstr, win_wchstr – retrieve complex character string (with rendition) | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int in_wchnstr(cchar_t *wchstr, int n); int in_wchstr(cchar_t *wchstr); int mvin_wchnstr(int y, int x, cchar_t *wchstr, int n); int mvin_wchstr(int y, int x, cchar_t *wchstr); int mvwin_wchnstr(WINDOW *win, int y, int x, cchar_t *wchstr, int n); int mvwin_wchstr(WINDOW *win, int y, int x, cchar_t *wchstr); int win_wchnstr(WINDOW *win, cchar_t *wchstr, int n); int win_wchstr(WINDOW *win, cchar_t *wchstr);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The <code>in_wchstr()</code> and <code>win_wchstr()</code> functions retrieve a complex character string (with rendition) starting at the current cursor position of the <code>stdscr</code> window and window <code>win</code>, respectively, and ending at the right margin. The <code>mvin_wchstr()</code> and <code>mvwin_wchstr()</code> functions retrieve a complex character string located at the position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <code>win</code>).</p> <p>The <code>in_wchnstr()</code>, <code>win_wchnstr()</code>, <code>mvin_wchnstr()</code>, and <code>mvwin_wchnstr()</code> functions retrieve at most <code>n</code> characters from the window <code>stdscr</code> and <code>win</code>, respectively. The former two functions retrieve the string, starting at the current cursor position; the latter two commands retrieve the string, starting at the position specified by the <code>x</code> and <code>y</code> parameters.</p> <p>The retrieved character string (with renditions) is stored in the object pointed to by <code>wcval</code>.</p> | | | | | | | | | | |
| PARAMETERS | <table><tr><td><code>wchstr</code></td><td>Is a pointer to an object where the retrieved complex character string can be stored.</td></tr><tr><td><code>n</code></td><td>Is the number of characters not to exceed when retrieving <code>wchstr</code>.</td></tr><tr><td><code>y</code></td><td>Is the <code>y</code> (row) coordinate of the starting position of the string to be retrieved.</td></tr><tr><td><code>x</code></td><td>Is the <code>x</code> (column) coordinate of the starting position of the string to be retrieved.</td></tr><tr><td><code>win</code></td><td>Is a pointer to the window in which the string is to be retrieved.</td></tr></table> | <code>wchstr</code> | Is a pointer to an object where the retrieved complex character string can be stored. | <code>n</code> | Is the number of characters not to exceed when retrieving <code>wchstr</code> . | <code>y</code> | Is the <code>y</code> (row) coordinate of the starting position of the string to be retrieved. | <code>x</code> | Is the <code>x</code> (column) coordinate of the starting position of the string to be retrieved. | <code>win</code> | Is a pointer to the window in which the string is to be retrieved. |
| <code>wchstr</code> | Is a pointer to an object where the retrieved complex character string can be stored. | | | | | | | | | | |
| <code>n</code> | Is the number of characters not to exceed when retrieving <code>wchstr</code> . | | | | | | | | | | |
| <code>y</code> | Is the <code>y</code> (row) coordinate of the starting position of the string to be retrieved. | | | | | | | | | | |
| <code>x</code> | Is the <code>x</code> (column) coordinate of the starting position of the string to be retrieved. | | | | | | | | | | |
| <code>win</code> | Is a pointer to the window in which the string is to be retrieved. | | | | | | | | | | |

in_wchstr(3XCURSES)

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO in_wch(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

inwstr(3CURSES)

NAME | curs_inwstr, inwstr, innwstr, winwstr, winnwstr, mvinwstr, mvinnwstr, mvwinwstr, mvwinnwstr – get a string of wchar_t characters from a curses window

SYNOPSIS | `cc [flag ...] file ... -lcurses [library ..]`
| `#include <curses.h>`
|
| `int inwstr(wchar_t *wstr);`
| `int innwstr(wchar_t *wstr, int n);`
| `int winwstr(WINDOW *win, wchar_t *wstr);`
| `int winnwstr(WINDOW *win, wchar_t *wstr, int n);`
| `int mvinwstr(int y, int x, wchar_t *wstr);`
| `int mvinnwstr(int y, int x, wchar_t *wstr, int n);`
| `int mvwinwstr(WINDOW *win, int y, int x, wchar_t *wstr);`
| `int mvwinnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);`

DESCRIPTION | These routines return the string of wchar_t characters in wstr starting at the current cursor position in the named window and ending at the right margin of the window. Attributes are stripped from the characters. The four functions with n as the last argument return the string at most n wchar_t characters long.

RETURN VALUES | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | curses(3CURSES), attributes(5)

NOTES | The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <wctype.h>.

Note that all routines except winnwstr() may be macros.

| | |
|--------------------|---|
| NAME | innwstr, inwstr, mvinnwstr, mvinwstr, mvwinnwstr, mvwinwstr, winnwstr, winwstr – retrieve a wide character string (without rendition) |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int innwstr(wchar_t *wstr, int n); int inwstr(wchar_t *wstr); int mvinnwstr(int y, int x, wchar_t *wstr, int n); int mvinwstr(int y, int x, wchar_t *wstr); int mvwinnwstr(WINDOW*win, int y, int x, wchar_t *wstr, int n); int mvwinwstr(WINDOW*win, int y, int x, wchar_t *wstr); int winwstr(WINDOW*win, wchar_t *wstr); int winnwstr(WINDOW*win, wchar_t *wstr, int n);</pre> |
| PARAMETERS | <p><i>wstr</i> Is a pointer to an object that can hold the retrieved multibyte character string.</p> <p><i>n</i> Is the number of characters not to exceed when retrieving <i>wstr</i>.</p> <p><i>y</i> Is the y (row) coordinate of the starting position of the string to be retrieved.</p> <p><i>x</i> Is the x (column) coordinate of the starting position of the string to be retrieved.</p> <p><i>win</i> Is a pointer to the window in which the string is to be retrieved.</p> |
| DESCRIPTION | <p>The <code>inwstr()</code> and <code>winwstr()</code> functions retrieve a wide character string (without attributes) starting at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively, and ending at the right margin. The <code>mvinwstr()</code> and <code>mvwinwstr()</code> functions retrieve a wide character string located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>innwstr()</code>, <code>winnwstr()</code>, <code>mvinnwstr()</code>, and <code>mvwinnwstr()</code> functions retrieve at most <i>n</i> characters from the window <code>stdscr</code> and <i>win</i>, respectively. The former two functions retrieve the string starting at the current cursor position; the latter two commands return the string, starting at the position specified by the <i>x</i> and <i>y</i> parameters.</p> <p>All these functions store the retrieved string in the object pointed to by <i>wstr</i>. They only store complete wide characters. If the area pointed to by <i>wstr</i> is not large enough to hold at least one character, these functions fail.</p> |

inwstr(3XCURSES)

Only the character portion of the character/rendition pair is returned. To return the complete character/rendition pair, use `win_wchstr(3XCURSES)`.

RETURN VALUES On success, the `inwstr()`, `mvinwstr()`, `mvwinwstr()`, and `winwstr()` functions return OK. Otherwise, they return ERR.

On success, the `innwstr()`, `mvinnwstr()`, `mvwinnwstr()`, and `winnwstr()` functions return the number of characters read into the string. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `in_wch(3XCURSES)`, `in_wchnstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | curs_initscr, initscr, newterm, endwin, isendwin, set_term, delscreen – curses screen initialization and manipulation routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> WINDOW *initscr(void); int endwin(void); int isendwin(void); SCREEN *newterm(char *<i>type</i>, FILE *<i>outfd</i>, FILE *<i>infd</i>); SCREEN *set_term(SCREEN *<i>new</i>); void delscreen(SCREEN * <i>sp</i>);</pre> |
| DESCRIPTION | <p><code>initscr()</code> is almost always the first routine that should be called (the exceptions are <code>slk_init()</code>, <code>filter()</code>, <code>ripoffline()</code>, <code>use_env()</code> and, for multiple-terminal applications, <code>newterm()</code>.) This determines the terminal type and initializes all curses data structures. <code>initscr()</code> also causes the first call to <code>refresh()</code> to clear the screen. If errors occur, <code>initscr()</code> writes an appropriate error message to standard error and exits; otherwise, a pointer is returned to <code>stdscr()</code>. If the program needs an indication of error conditions, <code>newterm()</code> should be used instead of <code>initscr()</code>; <code>initscr()</code> should only be called once per application.</p> <p>A program that outputs to more than one terminal should use the <code>newterm()</code> routine for each terminal instead of <code>initscr()</code>. A program that needs an indication of error conditions, so it can continue to run in a line-oriented mode if the terminal cannot support a screen-oriented program, would also use this routine. The routine <code>newterm()</code> should be called once for each terminal. It returns a variable of type <code>SCREEN *</code> which should be saved as a reference to that terminal. The arguments are the <i>type</i> of the terminal to be used in place of <code>\$TERM</code>, a file pointer for output to the terminal, and another file pointer for input from the terminal (if <i>type</i> is <code>NULL</code>, <code>\$TERM</code> will be used). The program must also call <code>endwin()</code> for each terminal being used before exiting from curses. If <code>newterm()</code> is called more than once for the same terminal, the first terminal referred to must be the last one for which <code>endwin()</code> is called.</p> <p>A program should always call <code>endwin()</code> before exiting or escaping from curses mode temporarily. This routine restores tty modes, moves the cursor to the lower left-hand corner of the screen and resets the terminal into the proper non-visual mode. Calling <code>refresh()</code> or <code>doupdate()</code> after a temporary escape causes the program to resume visual mode.</p> <p>The <code>isendwin()</code> routine returns <code>TRUE</code> if <code>endwin()</code> has been called without any subsequent calls to <code>wrefresh()</code>, and <code>FALSE</code> otherwise.</p> |

isendwin(3CURSES)

The `set_term()` routine is used to switch between different terminals. The screen reference `new` becomes the new current terminal. The previous terminal is returned by the routine. This is the only routine which manipulates `SCREEN` pointers; all other routines affect only the current terminal.

The `delscreen()` routine frees storage associated with the `SCREEN` data structure. The `endwin()` routine does not do this, so `delscreen()` should be called after `endwin()` if a particular `SCREEN` is no longer needed.

RETURN VALUES `endwin()` returns the integer `ERR` upon failure and `OK` upon successful completion.

Routines that return pointers always return `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_kernel(3CURSES)`, `curl_refresh(3CURSES)`, `curl_slk(3CURSES)`, `curl_util(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `initscr()` and `newterm()` may be macros.

| NAME | endwin, isendwin – restore initial terminal environment | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int endwin(void); bool isendwin(void);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>endwin()</code> function restores the terminal after Curses activity by at least restoring the saved shell terminsl mode, flushing any output to the terminal, and moving the cursor to the first column of the last line of the screen. Refreshing a window resumes program mode. The application must call <code>endwin()</code> for each terminal being used before exiting. If <code>newterm(3XCURSES)</code> is called more than once for the same terminal, the first screen created must be the last one for which <code>endwin()</code> is called.</p> <p>The <code>isendiwin()</code> function indicates whether or not a screen has been refreshed since the last call to <code>endwin()</code>.</p> | | | | | | |
| RETURN VALUES | <p>Upon successful completion, the <code>endwin()</code> function returns OK. Otherwise, it returns ERR.</p> <p>The <code>isendwin()</code> function returns TRUE if <code>endwin()</code> has been called without any subsequent refresh. Otherwise, it returns FALSE.</p> | | | | | | |
| ERRORS | Non errors are defined. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>doupdate(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>newterm(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

is_linetouched(3CURSES)

| NAME | curs_touch, touchwin, touchline, untouchwin, wtouchln, is_linetouched, is_wintouched – curses refresh control routines | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int touchwin(WINDOW *win); int touchline(WINDOW *win, int start, int count); int untouchwin(WINDOW *win); int wtouchln(WINDOW *win, int y, int n, int changed); int is_linetouched(WINDOW *win, int line); int is_wintouched(WINDOW *win);</pre> | | | | |
| DESCRIPTION | <p>The touchwin() and touchline() routines throw away all optimization information about which parts of the window have been touched, by pretending that the entire window has been drawn on. This is sometimes necessary when using overlapping windows, since a change to one window affects the other window, but the records of which lines have been changed in the other window do not reflect the change. The routine touchline() only pretends that <i>count</i> lines have been changed, beginning with line <i>start</i>.</p> <p>The untouchwin() routine marks all lines in the window as unchanged since the last call to wrefresh().</p> <p>The wtouchln() routine makes <i>n</i> lines in the window, starting at line <i>y</i>, look as if they have (<i>changed</i>=1) or have not (<i>changed</i>=0) been changed since the last call to wrefresh().</p> <p>The is_linetouched() and is_wintouched() routines return TRUE if the specified line/window was modified since the last call to wrefresh(); otherwise they return FALSE. In addition, is_linetouched() returns ERR if line is not valid for the given window.</p> | | | | |
| RETURN VALUES | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion, unless otherwise noted in the preceding routine descriptions. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curs_refresh(3CURSES), curses(3CURSES), attributes(5) | | | | |
| NOTES | The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>. | | | | |

`is_linetouched(3CURSES)`

Note that all routines except `wtouchln()` may be macros.

is_linetouched(3XCURSES)

| | | | | | | | | | | | | | | | |
|--------------------|---|------------|---|-------------|---|--------------|--|--------------|--|----------|---|----------|--|----------------|---|
| NAME | is_linetouched, is_wintouched, touchline, touchwin, untouchwin, wtouchln – control window refresh | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> bool is_linetouched(WINDOW *win, int line); bool is_wintouched(WINDOW *win); int touchline(WINDOW *win, int start, int count); int touchwin(WINDOW *win); int untouchwin(WINDOW *win); int wtouchln(WINDOW *win, int y, int n, int changed);</pre> | | | | | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>win</i></td><td>Is a pointer to the window in which the refresh is to be controlled or monitored.</td></tr><tr><td><i>line</i></td><td>Is the line to be checked for change since refresh.</td></tr><tr><td><i>start</i></td><td>Is the starting line number of the portion of the window to make appear changed.</td></tr><tr><td><i>count</i></td><td>Is the number of lines in the window to mark as changed.</td></tr><tr><td><i>y</i></td><td>Is the starting line number of the portion of the window to make appear changed or not changed.</td></tr><tr><td><i>n</i></td><td>Is the number of lines in the window to mark as changed.</td></tr><tr><td><i>changed</i></td><td>Is a flag indicating whether to make lines look changed (0) or not changed (1).</td></tr></table> | <i>win</i> | Is a pointer to the window in which the refresh is to be controlled or monitored. | <i>line</i> | Is the line to be checked for change since refresh. | <i>start</i> | Is the starting line number of the portion of the window to make appear changed. | <i>count</i> | Is the number of lines in the window to mark as changed. | <i>y</i> | Is the starting line number of the portion of the window to make appear changed or not changed. | <i>n</i> | Is the number of lines in the window to mark as changed. | <i>changed</i> | Is a flag indicating whether to make lines look changed (0) or not changed (1). |
| <i>win</i> | Is a pointer to the window in which the refresh is to be controlled or monitored. | | | | | | | | | | | | | | |
| <i>line</i> | Is the line to be checked for change since refresh. | | | | | | | | | | | | | | |
| <i>start</i> | Is the starting line number of the portion of the window to make appear changed. | | | | | | | | | | | | | | |
| <i>count</i> | Is the number of lines in the window to mark as changed. | | | | | | | | | | | | | | |
| <i>y</i> | Is the starting line number of the portion of the window to make appear changed or not changed. | | | | | | | | | | | | | | |
| <i>n</i> | Is the number of lines in the window to mark as changed. | | | | | | | | | | | | | | |
| <i>changed</i> | Is a flag indicating whether to make lines look changed (0) or not changed (1). | | | | | | | | | | | | | | |
| DESCRIPTION | <p>The touchwin() function marks the entire window as dirty. This makes it appear to X/Open Curses as if the whole window has been changed, thus causing the entire window to be rewritten with the next call to refresh(3XCURSES). This is sometimes necessary when using overlapping windows; the change to one window will not be reflected in the other and, hence will not be recorded.</p> <p>The touchline() function marks as dirty a portion of the window starting at line <i>start</i> and continuing for <i>count</i> lines instead of the entire window. Consequently, that portion of the window is updated with the next call to refresh().</p> <p>The untouchwin() function marks all lines in the window as unchanged since the last refresh, ensuring that it is not updated.</p> | | | | | | | | | | | | | | |

is_linetouched(3XCURSES)

The `wtouchln()` function marks n lines starting at line y as either changed ($changed=1$) or unchanged ($changed=0$) since the last refresh.

To find out which lines or windows have been changed since the last refresh, use the `is_linetouched()` and `is_wintouched()` functions, respectively. These return `TRUE` if the specified line or window have been changed since the last call to `refresh()` or `FALSE` if no changes have been made.

RETURN VALUES On success, these functions return `OK`. Otherwise, they return `ERR`.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `doupdate(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

is_wintouched(3CURSES)

| NAME | curs_touch, touchwin, touchline, untouchwin, wtouchln, is_linetouched, is_wintouched – curses refresh control routines | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int touchwin(WINDOW *win); int touchline(WINDOW *win, int start, int count); int untouchwin(WINDOW *win); int wtouchln(WINDOW *win, int y, int n, int changed); int is_linetouched(WINDOW *win, int line); int is_wintouched(WINDOW *win);</pre> | | | | |
| DESCRIPTION | <p>The touchwin() and touchline() routines throw away all optimization information about which parts of the window have been touched, by pretending that the entire window has been drawn on. This is sometimes necessary when using overlapping windows, since a change to one window affects the other window, but the records of which lines have been changed in the other window do not reflect the change. The routine touchline() only pretends that <i>count</i> lines have been changed, beginning with line <i>start</i>.</p> <p>The untouchwin() routine marks all lines in the window as unchanged since the last call to wrefresh().</p> <p>The wtouchln() routine makes <i>n</i> lines in the window, starting at line <i>y</i>, look as if they have (<i>changed</i>=1) or have not (<i>changed</i>=0) been changed since the last call to wrefresh().</p> <p>The is_linetouched() and is_wintouched() routines return TRUE if the specified line/window was modified since the last call to wrefresh(); otherwise they return FALSE. In addition, is_linetouched() returns ERR if line is not valid for the given window.</p> | | | | |
| RETURN VALUES | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion, unless otherwise noted in the preceding routine descriptions. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curs_refresh(3CURSES), curses(3CURSES), attributes(5) | | | | |
| NOTES | The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>. | | | | |

`is_wintouched(3CURSES)`

Note that all routines except `wtouchln()` may be macros.

is_wintouched(3XCURSES)

| | | | | | | | | | | | | | | | |
|--------------------|--|------------|---|-------------|---|--------------|--|--------------|--|----------|---|----------|--|----------------|---|
| NAME | <code>is_linetouched</code> , <code>is_wintouched</code> , <code>touchline</code> , <code>touchwin</code> , <code>untouchwin</code> , <code>wtouchln</code> – control window refresh | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> bool is_linetouched(WINDOW *win, int line); bool is_wintouched(WINDOW *win); int touchline(WINDOW *win, int start, int count); int touchwin(WINDOW *win); int untouchwin(WINDOW *win); int wtouchln(WINDOW *win, int y, int n, int changed);</pre> | | | | | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>win</i></td><td>Is a pointer to the window in which the refresh is to be controlled or monitored.</td></tr><tr><td><i>line</i></td><td>Is the line to be checked for change since refresh.</td></tr><tr><td><i>start</i></td><td>Is the starting line number of the portion of the window to make appear changed.</td></tr><tr><td><i>count</i></td><td>Is the number of lines in the window to mark as changed.</td></tr><tr><td><i>y</i></td><td>Is the starting line number of the portion of the window to make appear changed or not changed.</td></tr><tr><td><i>n</i></td><td>Is the number of lines in the window to mark as changed.</td></tr><tr><td><i>changed</i></td><td>Is a flag indicating whether to make lines look changed (0) or not changed (1).</td></tr></table> | <i>win</i> | Is a pointer to the window in which the refresh is to be controlled or monitored. | <i>line</i> | Is the line to be checked for change since refresh. | <i>start</i> | Is the starting line number of the portion of the window to make appear changed. | <i>count</i> | Is the number of lines in the window to mark as changed. | <i>y</i> | Is the starting line number of the portion of the window to make appear changed or not changed. | <i>n</i> | Is the number of lines in the window to mark as changed. | <i>changed</i> | Is a flag indicating whether to make lines look changed (0) or not changed (1). |
| <i>win</i> | Is a pointer to the window in which the refresh is to be controlled or monitored. | | | | | | | | | | | | | | |
| <i>line</i> | Is the line to be checked for change since refresh. | | | | | | | | | | | | | | |
| <i>start</i> | Is the starting line number of the portion of the window to make appear changed. | | | | | | | | | | | | | | |
| <i>count</i> | Is the number of lines in the window to mark as changed. | | | | | | | | | | | | | | |
| <i>y</i> | Is the starting line number of the portion of the window to make appear changed or not changed. | | | | | | | | | | | | | | |
| <i>n</i> | Is the number of lines in the window to mark as changed. | | | | | | | | | | | | | | |
| <i>changed</i> | Is a flag indicating whether to make lines look changed (0) or not changed (1). | | | | | | | | | | | | | | |
| DESCRIPTION | <p>The <code>touchwin()</code> function marks the entire window as dirty. This makes it appear to X/Open Curses as if the whole window has been changed, thus causing the entire window to be rewritten with the next call to <code>refresh(3XCURSES)</code>. This is sometimes necessary when using overlapping windows; the change to one window will not be reflected in the other and, hence will not be recorded.</p> <p>The <code>touchline()</code> function marks as dirty a portion of the window starting at line <i>start</i> and continuing for <i>count</i> lines instead of the entire window. Consequently, that portion of the window is updated with the next call to <code>refresh()</code>.</p> <p>The <code>untouchwin()</code> function marks all lines in the window as unchanged since the last refresh, ensuring that it is not updated.</p> | | | | | | | | | | | | | | |

is_wintouched(3XCURSES)

The `wtouchln()` function marks *n* lines starting at line *y* as either changed (*changed=1*) or unchanged (*changed=0*) since the last refresh.

To find out which lines or windows have been changed since the last refresh, use the `is_linetouched()` and `is_wintouched()` functions, respectively. These return `TRUE` if the specified line or window have been changed since the last call to `refresh()` or `FALSE` if no changes have been made.

RETURN VALUES On success, these functions return `OK`. Otherwise, they return `ERR`.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `doupdate(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

item_count(3CURSES)

| NAME | menu_items, set_menu_items, item_count – connect and disconnect items to and from menus | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int set_menu_items(MENU *menu, ITEM **items); ITEM **menu_items(MENU *menu); int item_count(MENU *menu);</pre> | | | | |
| DESCRIPTION | set_menu_items() changes the item pointer array connected to <i>menu</i> to the item pointer array <i>items</i> . menu_items() returns a pointer to the item pointer array connected to <i>menu</i> . item_count() returns the number of items in <i>menu</i> . | | | | |
| RETURN VALUES | menu_items() returns NULL on error. item_count() returns -1 on error. set_menu_items() returns one of the following: E_OK The routine returned successfully. E_SYSTEM_ERROR System error. E_BAD_ARGUMENT An incorrect argument was passed to the routine. E_POSTED The menu is already posted. E_CONNECTED One or more items are already connected to another menu. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), menus(3CURSES), attributes(5) | | | | |
| NOTES | The header <menu.h> automatically includes the headers <eti.h> and <curses.h>. | | | | |

item_description(3CURSES)

NAME menu_item_name, item_name, item_description – get menu item name and description

SYNOPSIS

```
cc [ flag ... ] file ... -lmenu -lcurses [ library .. ]
#include <menu.h>
```

```
char *item_name (ITEM *item) ;
char *item_description (ITEM *item) ;
```

DESCRIPTION item_name() returns a pointer to the name of *item*.

item_description() returns a pointer to the description of *item*.

RETURN VALUES These routines return NULL on error.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), menu_new(3CURSES), attributes(5)

NOTES The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

item_index(3CURSES)

NAME menu_item_current, set_current_item, current_item, set_top_row, top_row, item_index
– set and get current menus items

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

int set_current_item(MENU *menu, ITEM *item);
ITEM *current_item(MENU *menu);
int set_top_row(MENU *menu, int row);
int top_row(MENU *menu);
int item_index(ITEM *item);
```

DESCRIPTION The current item of a menu is the item where the cursor is currently positioned. `set_current_item()` sets the current item of *menu* to *item*. `current_item()` returns a pointer to the the current item in *menu*.

`set_top_row()` sets the top row of *menu* to *row*. The left-most item on the new top row becomes the current item. `top_row()` returns the number of the menu row currently displayed at the top of *menu*.

`item_index()` returns the index to the *item* in the item pointer array. The value of this index ranges from 0 through *N*-1, where *N* is the total number of items connected to the menu.

RETURN VALUES `current_item()` returns NULL on error.

`top_row()` and `index_item()` return -1 on error.

`set_current_item()` and `set_top_row()` return one of the following:

| | |
|-----------------|--|
| E_OK | The routine returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. |
| E_BAD_STATE | The routine was called from an initialization or termination function. |
| E_NOT_CONNECTED | No items are connected to the menu. |

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3CURSES)`, `menus(3CURSES)`, `attributes(5)`

item_index(3CURSES)

NOTES The header `<menu.h>` automatically includes the headers `<eti.h>` and `< curses.h>`.

item_init(3CURSES)

| | | | | | |
|----------------------|--|------|------------------------------------|----------------|---------------|
| NAME | menu_hook, set_item_init, item_init, set_item_term, item_term, set_menu_init, menu_init, set_menu_term, menu_term – assign application-specific routines for automatic invocation by menus | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int set_item_init(MENU *menu, void (*func)(MENU *)); int set_item_term(MENU *menu, void (*func)(MENU *)); void item_term(MENU *menu); int set_menu_init(MENU *menu, void (*func)(MENU *)); void menu_init(MENU *menu); int set_menu_term(MENU *menu, void (*func)(MENU *)); void menu_term(MENU *menu);</pre> | | | | |
| DESCRIPTION | <p>set_item_init() assigns the application-defined function to be called when the <i>menu</i> is posted and just after the current item changes. item_init() returns a pointer to the item initialization routine, if any, called when the <i>menu</i> is posted and just after the current item changes.</p> <p>set_item_term() assigns an application-defined function to be called when the <i>menu</i> is unposted and just before the current item changes. item_term() returns a pointer to the termination function, if any, called when the <i>menu</i> is unposted and just before the current item changes.</p> <p>set_menu_init() assigns an application-defined function to be called when the <i>menu</i> is posted and just after the top row changes on a posted menu. menu_init() returns a pointer to the menu initialization routine, if any, called when the <i>menu</i> is posted and just after the top row changes on a posted menu.</p> <p>set_menu_term() assigns an application-defined function to be called when the <i>menu</i> is unposted and just before the top row changes on a posted menu. menu_term() returns a pointer to the menu termination routine, if any, called when the <i>menu</i> is unposted and just before the top row changes on a posted menu.</p> | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <table><tr><td>E_OK</td><td>The routine returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr></table> | E_OK | The routine returned successfully. | E_SYSTEM_ERROR | System error. |
| E_OK | The routine returned successfully. | | | | |
| E_SYSTEM_ERROR | System error. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |

item_init(3CURSES)

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), attributes(5)

NOTES The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

item_name(3CURSES)

- NAME** menu_item_name, item_name, item_description – get menus item name and description
- SYNOPSIS**

```
cc [ flag ... ] file ... -lmenu -lcurses [ library .. ]
#include <menu.h>

char *item_name (ITEM *item) ;
char *item_description (ITEM *item) ;
```
- DESCRIPTION** item_name() returns a pointer to the name of *item*.
item_description() returns a pointer to the description of *item*.
- RETURN VALUES** These routines return NULL on error.
- ATTRIBUTES** See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), menu_new(3CURSES), attributes(5)

NOTES The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

| NAME | menu_item_opts, set_item_opts, item_opts_on, item_opts_off, item_opts – menus item option routines | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int set_item_opts (ITEM *item, OPTIONS opts); int item_opts_on (ITEM *item, OPTIONS opts); int item_opts_off (ITEM *item, OPTIONS opts); OPTIONS item_opts (ITEM *item);</pre> | | | | |
| DESCRIPTION | <p>set_item_opts() turns on the named options for <i>item</i> and turns off all other options. Options are boolean values that can be OR-ed together.</p> <p>item_opts_on() turns on the named options for <i>item</i>; no other option is changed.</p> <p>item_opts_off() turns off the named options for <i>item</i>; no other option is changed.</p> <p>item_opts() returns the current options of <i>item</i>.</p> <p>O_SELECTABLE The item can be selected during menu processing.</p> | | | | |
| RETURN VALUES | <p>Except for item_opts(), these routines return one of the following:</p> <p>E_OK The routine returned successfully.</p> <p>E_SYSTEM_ERROR System error.</p> | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), menus(3CURSES), attributes(5) | | | | |
| NOTES | The header <menu.h> automatically includes the headers <eti.h> and <curses.h>. | | | | |

item_opts_off(3CURSES)

| NAME | menu_item_opts, set_item_opts, item_opts_on, item_opts_off, item_opts – menus item option routines | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int set_item_opts (ITEM *item, OPTIONS opts); int item_opts_on (ITEM *item, OPTIONS opts); int item_opts_off (ITEM *item, OPTIONS opts); OPTIONS item_opts (ITEM *item);</pre> | | | | |
| DESCRIPTION | <p>set_item_opts() turns on the named options for <i>item</i> and turns off all other options. Options are boolean values that can be OR-ed together.</p> <p>item_opts_on() turns on the named options for <i>item</i>; no other option is changed.</p> <p>item_opts_off() turns off the named options for <i>item</i>; no other option is changed.</p> <p>item_opts() returns the current options of <i>item</i>.</p> <p>O_SELECTABLE The item can be selected during menu processing.</p> | | | | |
| RETURN VALUES | <p>Except for item_opts(), these routines return one of the following:</p> <p>E_OK The routine returned successfully.</p> <p>E_SYSTEM_ERROR System error.</p> | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), menus(3CURSES), attributes(5) | | | | |
| NOTES | The header <menu.h> automatically includes the headers <eti.h> and <curses.h>. | | | | |

| NAME | menu_item_opts, set_item_opts, item_opts_on, item_opts_off, item_opts – menus item option routines | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int set_item_opts (ITEM *item, OPTIONS opts); int item_opts_on (ITEM *item, OPTIONS opts); int item_opts_off (ITEM *item, OPTIONS opts); OPTIONS item_opts (ITEM *item);</pre> | | | | |
| DESCRIPTION | <p>set_item_opts() turns on the named options for <i>item</i> and turns off all other options. Options are boolean values that can be OR-ed together.</p> <p>item_opts_on() turns on the named options for <i>item</i>; no other option is changed.</p> <p>item_opts_off() turns off the named options for <i>item</i>; no other option is changed.</p> <p>item_opts() returns the current options of <i>item</i>.</p> <p>O_SELECTABLE The item can be selected during menu processing.</p> | | | | |
| RETURN VALUES | <p>Except for item_opts(), these routines return one of the following:</p> <p>E_OK The routine returned successfully.</p> <p>E_SYSTEM_ERROR System error.</p> | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), menus(3CURSES), attributes(5) | | | | |
| NOTES | The header <menu.h> automatically includes the headers <eti.h> and <curses.h>. | | | | |

item_term(3CURSES)

| | | | | | |
|----------------------|--|------|------------------------------------|----------------|---------------|
| NAME | menu_hook, set_item_init, item_init, set_item_term, item_term, set_menu_init, menu_init, set_menu_term, menu_term – assign application-specific routines for automatic invocation by menus | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int set_item_init(MENU *menu, void (*func)(MENU *)); int set_item_term(MENU *menu, void (*func)(MENU *)); void item_term(MENU *menu); int set_menu_init(MENU *menu, void (*func)(MENU *)); void menu_init(MENU *menu); int set_menu_term(MENU *menu, void (*func)(MENU *)); void menu_term(MENU *menu);</pre> | | | | |
| DESCRIPTION | <p>set_item_init() assigns the application-defined function to be called when the <i>menu</i> is posted and just after the current item changes. item_init() returns a pointer to the item initialization routine, if any, called when the <i>menu</i> is posted and just after the current item changes.</p> <p>set_item_term() assigns an application-defined function to be called when the <i>menu</i> is unposted and just before the current item changes. item_term() returns a pointer to the termination function, if any, called when the <i>menu</i> is unposted and just before the current item changes.</p> <p>set_menu_init() assigns an application-defined function to be called when the <i>menu</i> is posted and just after the top row changes on a posted menu. menu_init() returns a pointer to the menu initialization routine, if any, called when the <i>menu</i> is posted and just after the top row changes on a posted menu.</p> <p>set_menu_term() assigns an application-defined function to be called when the <i>menu</i> is unposted and just before the top row changes on a posted menu. menu_term() returns a pointer to the menu termination routine, if any, called when the <i>menu</i> is unposted and just before the top row changes on a posted menu.</p> | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <table><tr><td>E_OK</td><td>The routine returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr></table> | E_OK | The routine returned successfully. | E_SYSTEM_ERROR | System error. |
| E_OK | The routine returned successfully. | | | | |
| E_SYSTEM_ERROR | System error. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |

item_term(3CURSES)

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), attributes(5)

NOTES The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

item_userptr(3CURSES)

| NAME | menu_item_userptr, set_item_userptr, item_userptr – associate application data with menus items | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int set_item_userptr(ITEM *item, char *userptr); char *item_userptr(ITEM *item);</pre> | | | | |
| DESCRIPTION | Every item has an associated user pointer that can be used to store relevant information. set_item_userptr() sets the user pointer of item. item_userptr() returns the user pointer of item. | | | | |
| RETURN VALUES | item_userptr() returns NULL on error. set_item_userptr() returns one of the following: E_OK The routine returned successfully. E_SYSTEM_ERROR System error. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: <table border="1" data-bbox="444 940 1414 1031"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), menus(3CURSES), attributes(5) | | | | |
| NOTES | The header <menu.h> automatically includes the headers <eti.h> and <curses.h>. | | | | |

item_value(3CURSES)

NAME menu_item_value, set_item_value, item_value – set and get menus item values

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

int set_item_value (ITEM *item, int bool);
int item_value (ITEM *item);
```

DESCRIPTION Unlike single-valued menus, multi-valued menus enable the end-user to select one or more items from a menu. `set_item_value()` sets the selected value of the *item* — TRUE (selected) or FALSE (not selected). `set_item_value()` may be used only with multi-valued menus. To make a menu multi-valued, use `set_menu_opts` or `menu_opts_off()` to turn off the option `O_ONEVALUE`. (See `menu_opts(3CURSES)`).

`item_value()` returns the select value of *item*, either TRUE (selected) or FALSE (unselected).

RETURN VALUES `set_item_value()` returns one of the following:

- `E_OK` The routine returned successfully.
- `E_SYSTEM_ERROR` System error.
- `E_REQUEST_DENIED` The menu driver could not process the request.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3CURSES)`, `menus(3CURSES)`, `menu_opts(3CURSES)`, `attributes(5)`

NOTES The header `<menu.h>` automatically includes the headers `<eti.h>` and `<curses.h>`.

item_visible(3CURSES)

- NAME** menu_item_visible, item_visible – tell if menu item is visible
- SYNOPSIS**

```
cc [ flag ... ] file ... -lmenu -lcurses [ library .. ]
#include <menu.h>

int item_visible (ITEM *item);
```
- DESCRIPTION** A menu item is visible if it currently appears in the subwindow of a posted menu. item_visible() returns TRUE if *item* is visible, otherwise it returns FALSE.
- ATTRIBUTES** See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), menu_new(3CURSES), attributes(5)

NOTES The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

| | |
|--------------------|--|
| NAME | curs_util, unctrl, keyname, filter, use_env, putwin, getwin, delay_output, flushinp – curses miscellaneous utility routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> char *unctrl(chtype c); char *keyname(int c); int filter(void); void use_env(char bool); int putwin(WINDOW *win, FILE *filep); WINDOW *getwin(FILE *filep); int delay_output(int ms); int flushinp(void);</pre> |
| DESCRIPTION | <p>The <code>unctrl()</code> macro expands to a character string which is a printable representation of the character <code>c</code>. Control characters are displayed in the <code>^X</code> notation. Printing characters are displayed as is.</p> <p>With the <code>keyname()</code> routine, a character string corresponding to the key <code>c</code> is returned.</p> <p>The <code>filter()</code> routine, if used, is called before <code>initscr()</code> or <code>newterm()</code> are called. It makes <code>curses</code> think that there is a one-line screen. <code>curses</code> does not use any terminal capabilities that assume that they know on what line of the screen the cursor is positioned.</p> <p>The <code>use_env()</code> routine, if used, is called before <code>initscr()</code> or <code>newterm()</code> are called. When called with <code>FALSE</code> as an argument, the values of <code>lines</code> and <code>columns</code> specified in the <code>terminfo</code> database will be used, even if environment variables <code>LINES</code> and <code>COLUMNS</code> (used by default) are set, or if <code>curses</code> is running in a window (in which case default behavior would be to use the window size if <code>LINES</code> and <code>COLUMNS</code> are not set).</p> <p>With the <code>putwin()</code> routine, all data associated with window <code>win</code> is written into the file to which <code>filep</code> points. This information can be later retrieved using the <code>getwin()</code> function.</p> <p>The <code>getwin()</code> routine reads window related data stored in the file by <code>putwin()</code>. The routine then creates and initializes a new window using that data. It returns a pointer to the new window.</p> <p>The <code>delay_output()</code> routine inserts an <code>ms</code> millisecond pause in output. This routine should not be used extensively because padding characters are used rather than a CPU pause.</p> <p>The <code>flushinp()</code> routine throws away any typeahead that has been typed by the user and has not yet been read by the program.</p> |

keyname(3CURSES)

RETURN VALUES Except for `flushinp()`, routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`flushinp()` always returns `OK`.

Routines that return pointers return `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_initscr(3CURSES)`, `curs_scr_dump(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `unctrl()` is a macro, which is defined in `<unctrl.h>`.

| NAME | keyname, key_name – return character string used as key name | | | | | | | | | | | | |
|---|--|-------|--------------------|-------------------|--------------------|-------------------|----|---|-----|---|----------|-------------------|-------------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> char *keyname(int c); char *key_name(wchar_t wc);</pre> | | | | | | | | | | | | |
| PARAMETERS | <p><i>c</i> Is an 8 bit-character or a key code.</p> <p><i>wc</i> Is a wide character key name.</p> | | | | | | | | | | | | |
| DESCRIPTION | <p>The <code>keyname()</code> function returns a string pointer to the key name. Make a duplicate copy of the returned string if you plan to modify it.</p> <p>The <code>key_name()</code> function is similar except that it accepts a wide character key name.</p> <p>The following table shows the format of the key name based on the input.</p> <table border="1" data-bbox="461 953 1430 1285"> <thead> <tr> <th>Input</th> <th>Format of Key Name</th> </tr> </thead> <tbody> <tr> <td>Visible character</td> <td>The same character</td> </tr> <tr> <td>Control character</td> <td>^X</td> </tr> <tr> <td>Meta-character (<code>keyname()</code> only)</td> <td>M-X</td> </tr> <tr> <td>Key value defined in <code><curses.h></code> (<code>keyname()</code> only)</td> <td>KEY_name</td> </tr> <tr> <td>None of the above</td> <td>UNKNOWN KEY</td> </tr> </tbody> </table> <p>In the preceding table, <i>X</i> can be either a visible character with the high bit cleared or a control character.</p> | Input | Format of Key Name | Visible character | The same character | Control character | ^X | Meta-character (<code>keyname()</code> only) | M-X | Key value defined in <code><curses.h></code> (<code>keyname()</code> only) | KEY_name | None of the above | UNKNOWN KEY |
| Input | Format of Key Name | | | | | | | | | | | | |
| Visible character | The same character | | | | | | | | | | | | |
| Control character | ^X | | | | | | | | | | | | |
| Meta-character (<code>keyname()</code> only) | M-X | | | | | | | | | | | | |
| Key value defined in <code><curses.h></code> (<code>keyname()</code> only) | KEY_name | | | | | | | | | | | | |
| None of the above | UNKNOWN KEY | | | | | | | | | | | | |
| RETURN VALUES | On success, these functions return a pointer to the string used as the key's name. Otherwise, they return a null pointer. | | | | | | | | | | | | |
| ERRORS | None. | | | | | | | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | | | | | | | |

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |

key_name(3XCURSES)

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `libcurses(3XCURSES)`, `meta(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | keyname, key_name – return character string used as key name | | | | | | | | | | | | |
|---|--|-------|--------------------|-------------------|--------------------|-------------------|----|---|-----|---|----------|-------------------|-------------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> char *keyname(int c); char *key_name(wchar_t wc);</pre> | | | | | | | | | | | | |
| PARAMETERS | <p><i>c</i> Is an 8 bit-character or a key code.</p> <p><i>wc</i> Is a wide character key name.</p> | | | | | | | | | | | | |
| DESCRIPTION | <p>The <code>keyname()</code> function returns a string pointer to the key name. Make a duplicate copy of the returned string if you plan to modify it.</p> <p>The <code>key_name()</code> function is similar except that it accepts a wide character key name.</p> <p>The following table shows the format of the key name based on the input.</p> <table border="1" data-bbox="461 953 1430 1285"> <thead> <tr> <th>Input</th> <th>Format of Key Name</th> </tr> </thead> <tbody> <tr> <td>Visible character</td> <td>The same character</td> </tr> <tr> <td>Control character</td> <td>^X</td> </tr> <tr> <td>Meta-character (<code>keyname()</code> only)</td> <td>M-X</td> </tr> <tr> <td>Key value defined in <code><curses.h></code> (<code>keyname()</code> only)</td> <td>KEY_name</td> </tr> <tr> <td>None of the above</td> <td>UNKNOWN KEY</td> </tr> </tbody> </table> <p>In the preceding table, <i>X</i> can be either a visible character with the high bit cleared or a control character.</p> | Input | Format of Key Name | Visible character | The same character | Control character | ^X | Meta-character (<code>keyname()</code> only) | M-X | Key value defined in <code><curses.h></code> (<code>keyname()</code> only) | KEY_name | None of the above | UNKNOWN KEY |
| Input | Format of Key Name | | | | | | | | | | | | |
| Visible character | The same character | | | | | | | | | | | | |
| Control character | ^X | | | | | | | | | | | | |
| Meta-character (<code>keyname()</code> only) | M-X | | | | | | | | | | | | |
| Key value defined in <code><curses.h></code> (<code>keyname()</code> only) | KEY_name | | | | | | | | | | | | |
| None of the above | UNKNOWN KEY | | | | | | | | | | | | |
| RETURN VALUES | On success, these functions return a pointer to the string used as the key's name. Otherwise, they return a null pointer. | | | | | | | | | | | | |
| ERRORS | None. | | | | | | | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | | | | | | | |

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |

keyname(3XCURSES)

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `libcurses(3XCURSES)`, `meta(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|---|
| NAME | curs_inopts, cbreak, nocbreak, echo, noecho, halfdelay, intrflush, keypad, meta, nodelay, notimeout, raw, noraw, noqiflush, qiflush, timeout, wtimeout, typeahead – curses terminal input option control routines |
| SYNOPSIS | <pre>cc [<i>flag ...</i>] <i>file ...</i> -lcurses [<i>library ...</i>] #include <curses.h> int cbreak(void); int nocbreak(void); int echo(void); int noecho(void); int halfdelay(int <i>tenths</i>); int intrflush(WINDOW *<i>win</i>, bool <i>bf</i>); int keypad(WINDOW *<i>win</i>, bool <i>bf</i>); int meta(WINDOW *<i>win</i>, bool <i>bf</i>); int nodelay(WINDOW *<i>win</i>, bool <i>bf</i>); int notimeout(WINDOW *<i>win</i>, bool <i>bf</i>); int raw(void); int noraw(void); void noqiflush(void); void qiflush(void); void timeout(int <i>delay</i>); void wtimeout(WINDOW *<i>win</i>, int <i>delay</i>); int typeahead(int <i>filtes</i>);</pre> |
| DESCRIPTION | <p>The <code>cbreak()</code> and <code>nocbreak()</code> routines put the terminal into and out of <code>cbreak()</code> mode, respectively. In this mode, characters typed by the user are immediately available to the program, and erase/kill character-processing is not performed. When out of this mode, the tty driver buffers the typed characters until a newline or carriage return is typed. Interrupt and flow control characters are unaffected by this mode. Initially the terminal may or may not be in <code>cbreak()</code> mode, as the mode is inherited; therefore, a program should call <code>cbreak()</code> or <code>nocbreak()</code> explicitly. Most interactive programs using <code>curses</code> set the <code>cbreak()</code> mode.</p> <p>Note that <code>cbreak()</code> overrides <code>raw()</code>. (See <code>curs_getch(3CURSES)</code> for a discussion of how these routines interact with <code>echo()</code> and <code>noecho()</code>.)</p> <p>The <code>echo()</code> and <code>noecho()</code> routines control whether characters typed by the user are echoed by <code>getch()</code> as they are typed. Echoing by the tty driver is always disabled, but initially <code>getch()</code> is in <code>echo</code> mode, so characters typed are echoed. Authors of</p> |

keypad(3CURSES)

most interactive programs prefer to do their own echoing in a controlled area of the screen, or not to echo at all, so they disable echoing by calling `noecho()`. (See `curs_getch(3CURSES)` for a discussion of how these routines interact with `cbreak()` and `nocbreak()`.)

The `halfdelay()` routine is used for half-delay mode, which is similar to `cbreak()` mode in that characters typed by the user are immediately available to the program. However, after blocking for *tenths* tenths of seconds, `ERR` is returned if nothing has been typed. The value of *tenths* must be a number between 1 and 255. Use `nocbreak()` to leave half-delay mode.

If the `intrflush()` option is enabled, (*bf* is `TRUE`), when an interrupt key is pressed on the keyboard (interrupt, break, quit) all output in the tty driver queue will be flushed, giving the effect of faster response to the interrupt, but causing `curses` to have the wrong idea of what is on the screen. Disabling (*bf* is `FALSE`), the option prevents the flush. The default for the option is inherited from the tty driver settings. The window argument is ignored.

The `keypad()` option enables the keypad of the user's terminal. If enabled (*bf* is `TRUE`), the user can press a function key (such as an arrow key) and `wgetch()` returns a single value representing the function key, as in `KEY_LEFT`. If disabled (*bf* is `FALSE`), `curses` does not treat function keys specially and the program has to interpret the escape sequences itself. If the keypad in the terminal can be turned on (made to transmit) and off (made to work locally), turning on this option causes the terminal keypad to be turned on when `wgetch()` is called. The default value for `keypad` is false.

Initially, whether the terminal returns 7 or 8 significant bits on input depends on the control mode of the tty driver (see `termio(7I)`). To force 8 bits to be returned, invoke `meta(win, TRUE)`. To force 7 bits to be returned, invoke `meta(win, FALSE)`. The window argument, *win*, is always ignored. If the terminfo capabilities `smm` (`meta_on`) and `rmm` (`meta_off`) are defined for the terminal, `smm` is sent to the terminal when `meta(win, TRUE)` is called and `rmm` is sent when `meta(win, FALSE)` is called.

The `nodelay()` option causes `getch()` to be a non-blocking call. If no input is ready, `getch()` returns `ERR`. If disabled (*bf* is `FALSE`), `getch()` waits until a key is pressed.

While interpreting an input escape sequence, `wgetch()` sets a timer while waiting for the next character. If `notimeout(win, TRUE)` is called, then `wgetch()` does not set a timer. The purpose of the timeout is to differentiate between sequences received from a function key and those typed by a user.

With the `raw()` and `noraw()` routines, the terminal is placed into or out of raw mode. Raw mode is similar to `cbreak()` mode, in that characters typed are immediately passed through to the user program. The differences are that in raw mode, the interrupt, quit, suspend, and flow control characters are all passed through uninterpreted, instead of generating a signal. The behavior of the `BREAK` key depends on other bits in the tty driver that are not set by `curses`.

keypad(3CURSES)

When the `noqiflush()` routine is used, normal flush of input and output queues associated with the `INTR`, `QUIT` and `SUSP` characters will not be done (see `termio(7I)`). When `qiflush()` is called, the queues will be flushed when these control characters are read.

The `timeout()` and `wtimeout()` routines set blocking or non-blocking read for a given window. If *delay* is negative, blocking read is used (that is, waits indefinitely for input). If *delay* is zero, then non-blocking read is used (that is, read returns `ERR` if no input is waiting). If *delay* is positive, then read blocks for *delay* milliseconds, and returns `ERR` if there is still no input. Hence, these routines provide the same functionality as `nodelay()`, plus the additional capability of being able to block for only *delay* milliseconds (where *delay* is positive).

`curses` does “line-breakout optimization” by looking for typeahead periodically while updating the screen. If input is found, and it is coming from a `tty`, the current update is postponed until `refresh()` or `doupdate()` is called again. This allows faster response to commands typed in advance. Normally, the input `FILE` pointer passed to `newterm()`, or `stdin` in the case that `initscr()` was used, will be used to do this typeahead checking. The `typeahead()` routine specifies that the file descriptor *fildev* is to be used to check for typeahead instead. If *fildev* is `-1`, then no typeahead checking is done.

RETURN VALUES All routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_getch(3CURSES)`, `curs_initscr(3CURSES)`, `curses(3CURSES)`, `attributes(5)`, `termio(7I)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `echo()`, `noecho()`, `halfdelay()`, `intrflush()`, `meta()`, `nodelay()`, `notimeout()`, `noqiflush()`, `qiflush()`, `timeout()`, and `wtimeout()` may be macros.

keypad(3XCURSES)

| | |
|--------------------------|---|
| NAME | keypad – enable/disable keypad handling |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int keypad(WINDOW *win, bool bf);</pre> |
| PARAMETERS | <p><i>win</i> Is a pointer to the window in which to enable/disable keypad handling.</p> <p><i>bf</i> Is a Boolean expression.</p> |
| DESCRIPTION | <p>The keypad() function controls keypad translation. If <i>bf</i> is TRUE, keypad translation is enabled. If <i>bf</i> is FALSE, keypad translation is disabled. The initial state is FALSE.</p> <p>This function affects the behavior of any function that provides keyboard input.</p> <p>If the terminal in use requires a command to enable it to transmit distinctive codes when a function key is pressed, then after keypad translation is first enabled, the implementation transmits this command to the terminal before an affected input function tries to read any characters from that terminal.</p> <p>The Curses input model provides the following ways to obtain input from the keyboard:</p> |
| Keypad processing | <p>The application can enable or disable keypad translation by calling keypad(). When translation is enabled, Curses attempts to translate a sequence of terminal input that represents the pressing of a function into a single key code. When translation is disabled, Curses passes terminal input to the application without such translation, and any interpretation of the input as representing the pressing of a keypad key must be done by the application.</p> <p>The complete set of key codes for keypad keys that Curses can process is specified by the constants defined in <curses.h> whose names begin with "KEY_". Each terminal type described in the terminfo database may support some or all of these key codes. The terminfo database specifies the sequence of input characters from the terminal type that correspond to each key code.</p> <p>The Curses implementation cannot translate keypad keys on terminals where pressing the keys does not transmit a unique sequence.</p> <p>When translation is enabled and a character that could be the beginning of a function key (such as escape) is received, Curses notes the time and begins accumulating characters. If Curses receives additional characters that represent the processing of a keypad key within an unspecified interval from the time the character was received, then Curses converts this input to a key code for presentation to the application. If such characters are not received during this interval, translation of this input does not</p> |

occur and the individual characters are presented to the application separately. (Because Curses waits for this interval to accumulate a key code, many terminals experience a delay between the time a user presses the escape key and the time the escape key is returned to the application.)

In addition, No Timeout Mode provides that in any case where Curses has received part of a function key sequence, it waits indefinitely for the complete key sequence. The “unspecified interval” in the previous paragraph becomes infinite in No Timeout Mode. No Timeout Mode allows the use of function keys over slow communication lines. No Timeout Mode lets the user type the individual characters of a function key sequence, but also delays application response when the user types a character (not a function key) that begins a function key sequence. For this reason, in No Timeout Mode many terminals will appear to hang between the time a user presses the escape key and the time another key is pressed. No Timeout Mode is switchable by calling `notimeout(3XCURSES)`.

If any special characters (<backspace>, <carriage return>, <newline>, <tab>) are defined or redefined to be characters that are members of a function key sequence, then Curses will be unable to recognize and translate those function keys.

Several of the modes discussed below are described in terms of availability of input. If keypad translation is enabled, then input is not available once Curses has begun receiving a keypad sequence until the sequence is completely received or the interval has elapsed.

Input Mode

The following four mutually-specific Curses modes let the application control the effect of flow-control characters, the interrupt character, the erase character, and the kill character:

| Input Mode | Effect |
|--------------------------|---|
| Cooked Mode | This achieves normal line-at-a-time processing with all special characters handled outside the application. This achieves the same effect as canonical-mode input processing. The state of the <code>ISIG</code> and <code>IXON</code> flags are not changed upon entering this mode by calling <code>nocbreak(3XCURSES)</code> , and are set upon entering this mode by calling <code>noraw(3XCURSES)</code> . Erase and kill characters are supported from any supported locale, no matter the width of the character. |
| <code>cbreak</code> Mode | Characters typed by the user are immediately available to the application and Curses does not perform special processing on either the erase character or the kill character. An application can set <code>cbreak</code> mode to do its own line editing but to let the abort character be used to abort the task. This mode achieves the same effect as non-canonical-mode, Case B input processing (with <code>MIN</code> set to 1 and <code>ICRNL</code> cleared.) The state of the <code>ISIG</code> and <code>IXON</code> flags are not changed upon entering this mode. |

keypad(3XCURSES)

| Input Mode | Effect |
|-----------------|---|
| Half-Delay Mode | The effect is the same as <code>cbreak</code> , except that input functions wait until a character is available or an interval defined by the application elapses, whichever comes first. This mode achieves the same effect as non-canonical-mode, Case C input processing (with <code>TIME</code> set to the value specified by the application.) The state of the <code>ISIG</code> and <code>IXON</code> flags are not changed upon entering this mode. |
| Raw Mode | Raw mode gives the application maximum control over terminal input. The application sees each character as it is typed. This achieves the same effect as non-canonical mode, Case D input processing. The <code>ISIG</code> and <code>IXON</code> flags are cleared upon entering this mode. |

The terminal interface settings are reported when the process calls `initscr(3XCURSES)` or `newterm(3XCURSES)` to initialize Curses and restores these settings when `endwin(3XCURSES)` is called. The initial input mode for Curses operations is especially unless Enhanced Curses compliance, in which the initial mode is `cbreak` mode, is supported.

The behavior of the `BREAK` key depends on other bits in the display driver that are not set by Curses.

Delay Mode

Two mutually-exclusive delay modes specify how quickly certain Curses functions return to the application when there is no terminal input waiting when the function is called:

| | |
|----------|--|
| No Delay | The function fails. |
| Delay | The application waits until text is passed through to the application. If <code>cbreak</code> or Raw Mode is set, this is after one character. Otherwise, this is after the first <code><newline></code> character, end-of-line character, or end-of-file character. |

The effect of No Delay Mode on function key processing is unspecified.

Echo processing

Echo mode determines whether Curses echoes typed characters to the screen. The effect of Echo mode is analogous to the effect of the `ECHO` flag in the local mode field of the `termios` structure associated with the terminal device connected to the window. However, Curses always clears the `ECHO` flag when invoked, to inhibit the operating system from performing echoing. The method of echoing characters is not identical to the operating system's method of echoing characters, because Curses performs additional processing of terminal input.

If in Echo mode, Curses performs 's' own echoing. Any visible input character is stored in the current or specified window by the input function that the application called, at that window's cursor position, as though `addch(3XCURSES)` were called, with all consequent effects such as cursor movement and wrapping.

keypad(3XCURSES)

If not in Echo mode, any echoing of input must be performed by the application. Applications often perform their own echoing in a controlled area of the screen, or do not echo at all, so they disable Echo mode.

It may not be possible to turn off echo processing for synchronous and networked asynchronous terminals because echo processing is done directly by the terminals. Applications running on such terminals should be aware that any characters typed will appear on the screen at wherever the cursor is positioned.

RETURN VALUES Upon successful completion, the keypad() function returns OK. Otherwise, it returns ERR.

ERRORS No errors are defined.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO addch(3XCURSES), endwin(3XCURSES), getch(3XCURSES), initscr(3XCURSES), libcurses(3XCURSES), newterm(3XCURSES), nobreak(3XCURSES), noraw(3XCURSES), attributes(5), standards(5)

killchar(3CURSES)

| | |
|----------------------|---|
| NAME | <code> curs_termattrs, baudrate, erasechar, has_ic, has_il, killchar, longname, termattrs, termname</code> – curses environment query routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int baudrate(void); char erasechar(void); int has_ic(void); int has_il(void); char killchar(void); char *longname(void); chtype termattrs(void); char *termname(void);</pre> |
| DESCRIPTION | <p>The <code>baudrate()</code> routine returns the output speed of the terminal. The number returned is in bits per second, for example 9600, and is an integer.</p> <p>With the <code>erasechar()</code> routine, the user's current erase character is returned.</p> <p>The <code>has_ic()</code> routine is true if the terminal has insert- and delete-character capabilities.</p> <p>The <code>has_il()</code> routine is true if the terminal has insert- and delete-line capabilities, or can simulate them using scrolling regions. This might be used to determine if it would be appropriate to turn on physical scrolling using <code>scrollok()</code>.</p> <p>With the <code>killchar()</code> routine, the user's current line kill character is returned.</p> <p>The <code>longname()</code> routine returns a pointer to a static area containing a verbose description of the current terminal. The maximum length of a verbose description is 128 characters. It is defined only after the call to <code>initscr()</code> or <code>newterm()</code>. The area is overwritten by each call to <code>newterm()</code> and is not restored by <code>set_term()</code>, so the value should be saved between calls to <code>newterm()</code> if <code>longname()</code> is going to be used with multiple terminals.</p> <p>If a given terminal doesn't support a video attribute that an application program is trying to use, <code>curses</code> may substitute a different video attribute for it. The <code>termattrs()</code> function returns a logical OR of all video attributes supported by the terminal. This information is useful when a <code>curses</code> program needs complete control over the appearance of the screen.</p> <p>The <code>termname()</code> routine returns the value of the environment variable <code>TERM</code> (truncated to 14 characters).</p> |
| RETURN VALUES | <code>longname()</code> and <code>termname()</code> return <code>NULL</code> on error. |

killchar(3CURSES)

Routines that return an integer return ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_initscr(3CURSES)`, `curl_outopts(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `termattrs()` may be a macro.

killchar(3XCURSES)

| NAME | erasechar, eraseswchar, killchar, killwchar – return current ERASE or KILL characters | | | | | | |
|----------------------|--|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> char erasechar(void); int eraseswchar(wchar_t *ch); char killchar(void); int killwchar(wchar_t *ch);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>erasechar()</code> function returns the current ERASE character from the tty driver. This character is used to delete the previous character during keyboard input. The returned value can be used when including deletion capability in interactive programs.</p> <p>The <code>killchar()</code> function is similar to <code>erasechar()</code>. It returns the current KILL character.</p> <p>The <code>eraseswchar()</code> and <code>killwchar()</code> functions are similar to <code>erasechar()</code> and <code>killchar()</code> respectively, but store the ERASE or KILL character in the object pointed to by <i>ch</i>.</p> | | | | | | |
| PARAMETERS | <p><i>ch</i> Is a pointer to a location where a character may be stored.</p> | | | | | | |
| RETURN VALUES | <p>For <code>erasechar()</code> and <code>killchar()</code>, the terminal's current ERASE or KILL character is returned.</p> <p>On success, the <code>eraseswchar()</code> and <code>killwchar()</code> functions return OK. Otherwise, they return ERR.</p> | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Standard</td></tr><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>getch(3XCURSES)</code> , <code>getstr(3XCURSES)</code> , <code>get_wch(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

| NAME | erasechar, erasewchar, killchar, killwchar – return current ERASE or KILL characters | | | | | | |
|----------------------|--|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> char erasechar(void); int erasewchar(wchar_t *ch); char killchar(void); int killwchar(wchar_t *ch);</pre> | | | | | | |
| DESCRIPTION | <p>The erasechar() function returns the current ERASE character from the tty driver. This character is used to delete the previous character during keyboard input. The returned value can be used when including deletion capability in interactive programs.</p> <p>The killchar() function is similar to erasechar(). It returns the current KILL character.</p> <p>The erasewchar() and killwchar() functions are similar to erasechar() and killchar() respectively, but store the ERASE or KILL character in the object pointed to by <i>ch</i>.</p> | | | | | | |
| PARAMETERS | <i>ch</i> Is a pointer to a location where a character may be stored. | | | | | | |
| RETURN VALUES | <p>For erasechar() and killchar(), the terminal's current ERASE or KILL character is returned.</p> <p>On success, the erasewchar() and killwchar() functions return OK. Otherwise, they return ERR.</p> | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | getch(3XCURSES), getstr(3XCURSES), get_wch(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5) | | | | | | |

label(3PLOT)

| | |
|--------------------|--|
| NAME | plot, arc, box, circle, closepl, closevt, cont, erase, label, line, linemod, move, openpl, openvt, point, space – graphics interface |
| SYNOPSIS | <pre>cc [flag ...] file ... -lplot [library...] #include <plot.h> void arc(short x0, short y0, short x1, short y1, short x2, short y2); void box(short x0, short y0, short x1, short y1); void circle(short x, short y, short r); void closepl(); void closevt(); void cont(short x, short y); void erase(); void label(char *s); void line(short x0, short y0, short x1, short y1); void linemod(char *s); void move(short x, short y); void openpl(); void openvt(); void point(short x, short y); void space(short x0, short y0, short x1, short y1);</pre> |
| DESCRIPTION | <p>These functions generate graphics output for a set of output devices. The format of the output is dependent upon which link editor option is used when the program is compiled and linked (see Link Editor).</p> <p>The term "current point" refers to the current setting for the x and y coordinates.</p> <p>The <code>arc()</code> function specifies a circular arc. The coordinates $(x0, y0)$ specify the center of the arc. The coordinates $(x1, y1)$ specify the starting point of the arc. The coordinates $(x2, y2)$ specify the end point of the circular arc.</p> <p>The <code>box()</code> function specifies a rectangle with coordinates $(x0, y0)$, $(x0, y1)$, $(x1, y0)$, and $(x1, y1)$. The current point is set to $(x1, y1)$.</p> <p>The <code>circle()</code> function specifies a circle with a center at the coordinates (x, y) and a radius of r.</p> <p>The <code>closevt()</code> and <code>closepl()</code> functions flush the output.</p> <p>The <code>cont()</code> function specifies a line beginning at the current point and ending at the coordinates (x, y). The current point is set to (x, y).</p> |

The `erase()` function starts another frame of output.

The `label()` function places the null terminated string `s` so that the first character falls on the current point. The string is then terminated by a NEWLINE character.

The `line()` function draws a line starting at the coordinates $(x0, y0)$ and ending at the coordinates $(x1, y1)$. The current point is set to $(x1, y1)$.

The `linemod()` function specifies the style for drawing future lines. `s` may contain one of the following: `dotted`, `solid`, `longdashed`, `shortdashed`, or `dotdashed`.

The `move()` function sets the current point to the coordinates (x, y) .

The `openpl()` or `openvt()` function must be called to open the device before any other plot functions are called.

The `point()` function plots the point given by the coordinates (x, y) . The current point is set to (x, y) .

The `space()` function specifies the size of the plotting area. The plot will be reduced or enlarged as necessary to fit the area specified. The coordinates $(x0, y0)$ specify the lower left hand corner of the plotting area. The coordinates $(x1, y1)$ specify the upper right hand corner of the plotting area.

Link Editor Various flavors of these functions exist for different output devices. They are obtained by using the following `ld(1)` options:

```
-lplot          device-independent graphics stream on standard output in the
                 format described in plot(4B)
-l300           GSI 300 terminal
-l300s          GSI 300S terminal
-l4014          Tektronix 4014 terminal
-l450           GSI 450 terminal
-lvt0
```

FILES

```
/usr/lib/libplot.a
  archive library
/usr/lib/libplot.so.1
  shared object
/usr/lib/sparcv9/libplot.so.1
  64-bit shared object
/usr/lib/lib300.a
  archive library
/usr/lib/lib300.so.1
  shared object
```

label(3PLOT)

/usr/lib/sparcv9/lib300.so.1
64-bit shared object

/usr/lib/lib300s.a
archive library

/usr/lib/lib300s.so.1
shared object

/usr/lib/sparcv9/lib300s.so.1
64-bit shared object

/usr/lib/lib4014.a
archive library

/usr/lib/lib4014.so.1
shared object

/usr/lib/sparcv9/lib4014.so.1
64-bit shared object

/usr/lib/lib450.a
archive library

/usr/lib/lib450.so.1
shared object

/usr/lib/sparcv9/lib450.so.1
64-bit shared object

/usr/lib/libvt0.a
archive library

/usr/lib/libvt0.so.1
shared object

/usr/lib/sparcv9/libvt0.so.1
64-bit shared object

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `graph(1)`, `ld(1)`, `libplot(3LIB)`, `plot(4B)`, `attributes(5)`

| | |
|--------------------|---|
| NAME | curs_ouptoks, clearok, idlok, idcok, immedok, leaveok, setscreg, wsetscreg, scrollok, nl, nonl – curses terminal output option control routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int clearok(WINDOW *win, bool bf); int idlok(WINDOW *win, bool bf); void idcok(WINDOW *win, bool bf); void immedok(WINDOW *win, bool bf); int leaveok(WINDOW *win, bool bf); int setscreg(int top, int bot); int wsetscreg(WINDOW *win, int top, int bot); int scrollok(WINDOW *win, bool bf); int nl(void); int nonl(void);</pre> |
| DESCRIPTION | <p>These routines set options that deal with output within curses. All options are initially FALSE, unless otherwise stated. It is not necessary to turn these options off before calling <code>endwin()</code>.</p> <p>With the <code>clearok()</code> routine, if enabled (<i>bf</i> is TRUE), the next call to <code>wrefresh()</code> with this window will clear the screen completely and redraw the entire screen from scratch. This is useful when the contents of the screen are uncertain, or in some cases for a more pleasing visual effect. If the <i>win</i> argument to <code>clearok()</code> is the global variable <code>curscr()</code>, the next call to <code>wrefresh()</code> with any window causes the screen to be cleared and repainted from scratch.</p> <p>With the <code>idlok()</code> routine, if enabled (<i>bf</i> is TRUE), <code>curses</code> considers using the hardware insert/delete line feature of terminals so equipped. If disabled (<i>bf</i> is FALSE), <code>curses</code> very seldom uses this feature. (The insert/delete character feature is always considered.) This option should be enabled only if the application needs insert/delete line, for example, for a screen editor. It is disabled by default because insert/delete line tends to be visually annoying when used in applications where it isn't really needed. If insert/delete line cannot be used, <code>curses</code> redraws the changed portions of all lines.</p> <p>With the <code>idcok()</code> routine, if enabled (<i>bf</i> is TRUE), <code>curses</code> considers using the hardware insert/delete character feature of terminals so equipped. This is enabled by default.</p> <p>With the <code>immedok()</code> routine, if enabled (<i>bf</i> is TRUE), any change in the window image, such as the ones caused by <code>waddch()</code>, <code>wclrtoobot()</code>, <code>wscrl()</code>, etc., automatically cause a call to <code>wrefresh()</code>. However, it may degrade the performance considerably, due to repeated calls to <code>wrefresh()</code>. It is disabled by default.</p> |

leaveok(3CURSES)

Normally, the hardware cursor is left at the location of the window cursor being refreshed. The `leaveok()` option allows the cursor to be left wherever the update happens to leave it. It is useful for applications where the cursor is not used, since it reduces the need for cursor motions. If possible, the cursor is made invisible when this option is enabled.

The `setscrreg()` and `wsetscrreg()` routines allow the application programmer to set a software scrolling region in a window. *top* and *bot* are the line numbers of the top and bottom margin of the scrolling region. (Line 0 is the top line of the window.) If this option and `scrollok()` are enabled, an attempt to move off the bottom margin line causes all lines in the scrolling region to scroll up one line. Only the text of the window is scrolled. (Note that this has nothing to do with the use of a physical scrolling region capability in the terminal, like that in the VT100. If `idlok()` is enabled and the terminal has either a scrolling region or insert/delete line capability, they will probably be used by the output routines.)

The `scrollok()` option controls what happens when the cursor of a window is moved off the edge of the window or scrolling region, either as a result of a newline action on the bottom line, or typing the last character of the last line. If disabled, (*bf* is FALSE), the cursor is left on the bottom line. If enabled, (*bf* is TRUE), `wrefresh()` is called on the window, and the physical terminal and window are scrolled up one line. (Note that in order to get the physical scrolling effect on the terminal, it is also necessary to call `idlok()`.)

The `nl()` and `nonl()` routines control whether newline is translated into carriage return and linefeed on output, and whether return is translated into newline on input. Initially, the translations do occur. By disabling these translations using `nonl()`, `curses` is able to make better use of the linefeed capability, resulting in faster cursor motion.

RETURN VALUES `setscrreg()` and `wsetscrreg()` return OK upon success and ERR upon failure. All other routines that return an integer always return OK.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_addch(3CURSES)`, `curs_clear(3CURSES)`, `curs_initscr(3CURSES)`, `curs_refresh(3CURSES)`, `curs_scroll(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `clearok()`, `leaveok()`, `scrollok()`, `idcok()`, `nl()`, `nonl()`, and `setscrreg()` may be macros.

`leaveok(3CURSES)`

The `immedok()` routine is useful for windows that are used as terminal emulators.

leaveok(3XCURSES)

| | |
|--------------------|---|
| NAME | clearok, idlok, leaveok, scrollok, setscrreg, wsetscrreg – terminal output control functions |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int clearok(WINDOW *win, bool bf); int idlok(WINDOW *win, bool bf); int leaveok(WINDOW *win, bool bf); int scrollok(WINDOW *win, bool bf); int setscrreg(int top, int bot); int wsetscrreg(WINDOW *win, int top, int bot);</pre> |
| DESCRIPTION | <p>These functions set options that deal with the output within Curses functions.</p> <p>The <code>clearok()</code> function assigns the value of <i>bf</i> to an internal flag in the specified window that governs clearing of the screen during a refresh. If, during a refresh operation on the specified window, the flag in <i>curscr</i> is TRUE or the flag in the specified window is TRUE, <code>clearok()</code> clears the screen, redraws it in its entirety, and sets the flag to FALSE in <i>curscr</i> and in the specified window. The initial state is unspecified</p> <p>The <code>idlok()</code> function specifies whether the implementation may use the hardware insert-line, delete-line, and scroll features of terminals so equipped. If <i>bf</i> is TRUE, use of these features is enabled. If <i>bf</i> is FALSE, use of these features is disabled and lines are instead redrawn as required. The initial state is FALSE.</p> <p>The <code>leaveok()</code> function controls the cursor position after a refresh operation. If <i>bf</i> is TRUE, refresh operations on the specified window may leave the terminal's cursor at an arbitrary position. If <i>bf</i> is FALSE, then at the end of any refresh operation, the terminal's cursor is positioned at the cursor position contained in the specified window. The initial state is FALSE.</p> <p>The <code>scrollok()</code> function controls the use of scrolling. If <i>bf</i> is TRUE, then scrolling is enabled for the specified window. If <i>bf</i> is FALSE, scrolling is disabled for the specified window. The initial state is FALSE.</p> <p>The <code>setscrreg()</code> and <code>wsetscrreg()</code> functions define a software scrolling region in the current or specified window. The <i>top</i> and <i>bottom</i> arguments are the line numbers of the first and last line defining the scrolling region. (Line 0 is the top line of the window.) If this option and <code>scrollok()</code> are enabled, an attempt to move off the last line of the margin causes all lines in the scrolling region to scroll one line in the</p> |

leaveok(3XCURSES)

direction of the first line. Only characters in the window are scrolled. If a software scrolling region is set and `scrollok()` is not enabled, an attempt to move off the last line of the margin does not reposition any lines in the scrolling region.

PARAMETERS

- win* Is a pointer to a window.
- bf* Is a Boolean expression.
- top* Is the top line of the scrolling region (top of the window is line 0).
- bot* Is the bottom line of the scrolling region (top of the window is line 0).

RETURN VALUES

Upon successful completion, the `setscrreg()` and `wsetscrreg()` functions return OK. Otherwise, they return ERR.

The other functions always return OK.

ERRORS

No errors are defined.

USAGE

The only reason to enable the `idlok()` feature is to use scrolling to achieve the visual effect of motion of a partial window, such as for a screen editor. In other cases, the feature can be visually annoying.

The `leaveok()` option provides greater efficiency for applications that do not use the cursor.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO

`bkgdset(3XCURSES)`, `clear(3XCURSES)`, `doupdate(3XCURSES)`, `libcurses(3XCURSES)`, `scr1(3XCURSES)`, `attributes(5)`, `standards(5)`

libcurses(3XCURSES)

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----------------------------|---|---------------------|-------------------------|--------------------------|-------------------|--------------------|--------------------------|----------------------|--------------------------|-------------------------|--------------------|------------------------|-----------------------|----------------------|-----------------------|---------------------|----------------------|-----------------------|-----------------------|----------------------|-----------------------|----------------------|---------------------|----------------------|-----------------------|-------------------|-------------------|----------------------|---------------------|------------------------|---------------------|-------------------------|------------------|----------------------|-------------------------------|---------------------|--------------------|--------------------|----------------------|-----------------------|-----------------------|----------------------------|------------------------|
| NAME | libcurses – X/Open Curses library | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h></pre> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DESCRIPTION | Functions in this library provide a terminal-independent method of updating character screens with reasonable optimization, conforming to X/Open Curses, Issue 4, Version 2. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| INTERFACES | The shared object <code>libcurses.so.2</code> provides the public interfaces defined below. See <code>intro(3)</code> for additional information on shared object interfaces. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | <table><tr><td><code>COLORS</code></td><td><code>COLOR_PAIR</code></td></tr><tr><td><code>COLOR_PAIRS</code></td><td><code>COLS</code></td></tr><tr><td><code>LINES</code></td><td><code>PAIR_NUMBER</code></td></tr><tr><td><code>add_wch</code></td><td><code>add_wchnstr</code></td></tr><tr><td><code>add_wchstr</code></td><td><code>addch</code></td></tr><tr><td><code>addchnstr</code></td><td><code>addchstr</code></td></tr><tr><td><code>addnstr</code></td><td><code>addnwstr</code></td></tr><tr><td><code>addstr</code></td><td><code>addwstr</code></td></tr><tr><td><code>attr_get</code></td><td><code>attr_off</code></td></tr><tr><td><code>attr_on</code></td><td><code>attr_set</code></td></tr><tr><td><code>attroff</code></td><td><code>attron</code></td></tr><tr><td><code>attrset</code></td><td><code>baudrate</code></td></tr><tr><td><code>beep</code></td><td><code>bkgd</code></td></tr><tr><td><code>bkgdset</code></td><td><code>bkgdnd</code></td></tr><tr><td><code>bkgdndset</code></td><td><code>border</code></td></tr><tr><td><code>border_set</code></td><td><code>box</code></td></tr><tr><td><code>box_set</code></td><td><code>can_change_color</code></td></tr><tr><td><code>cbreak</code></td><td><code>chgat</code></td></tr><tr><td><code>clear</code></td><td><code>clearok</code></td></tr><tr><td><code>clrtobot</code></td><td><code>clrtoeol</code></td></tr><tr><td><code>color_content</code></td><td><code>color_set</code></td></tr></table> | <code>COLORS</code> | <code>COLOR_PAIR</code> | <code>COLOR_PAIRS</code> | <code>COLS</code> | <code>LINES</code> | <code>PAIR_NUMBER</code> | <code>add_wch</code> | <code>add_wchnstr</code> | <code>add_wchstr</code> | <code>addch</code> | <code>addchnstr</code> | <code>addchstr</code> | <code>addnstr</code> | <code>addnwstr</code> | <code>addstr</code> | <code>addwstr</code> | <code>attr_get</code> | <code>attr_off</code> | <code>attr_on</code> | <code>attr_set</code> | <code>attroff</code> | <code>attron</code> | <code>attrset</code> | <code>baudrate</code> | <code>beep</code> | <code>bkgd</code> | <code>bkgdset</code> | <code>bkgdnd</code> | <code>bkgdndset</code> | <code>border</code> | <code>border_set</code> | <code>box</code> | <code>box_set</code> | <code>can_change_color</code> | <code>cbreak</code> | <code>chgat</code> | <code>clear</code> | <code>clearok</code> | <code>clrtobot</code> | <code>clrtoeol</code> | <code>color_content</code> | <code>color_set</code> |
| <code>COLORS</code> | <code>COLOR_PAIR</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>COLOR_PAIRS</code> | <code>COLS</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>LINES</code> | <code>PAIR_NUMBER</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>add_wch</code> | <code>add_wchnstr</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>add_wchstr</code> | <code>addch</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>addchnstr</code> | <code>addchstr</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>addnstr</code> | <code>addnwstr</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>addstr</code> | <code>addwstr</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>attr_get</code> | <code>attr_off</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>attr_on</code> | <code>attr_set</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>attroff</code> | <code>attron</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>attrset</code> | <code>baudrate</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>beep</code> | <code>bkgd</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>bkgdset</code> | <code>bkgdnd</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>bkgdndset</code> | <code>border</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>border_set</code> | <code>box</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>box_set</code> | <code>can_change_color</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>cbreak</code> | <code>chgat</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>clear</code> | <code>clearok</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>clrtobot</code> | <code>clrtoeol</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>color_content</code> | <code>color_set</code> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | |
|---------------|----------------|
| copywin | cur_term |
| curs_set | curscr |
| def_prog_mode | def_shell_mode |
| del_curterm | delay_output |
| delch | deleteln |
| delscreen | delwin |
| derwin | doupdate |
| dupwin | echo |
| echo_wchar | echochar |
| endwin | erase |
| erasechar | eraseswchar |
| filter | flash |
| flushinp | get_wch |
| get_wstr | getbkgd |
| getbkgrnd | getcchar |
| getch | getn_wstr |
| getnstr | getstr |
| getwin | halfdelay |
| has_colors | has_ic |
| has_il | hline |
| hline_set | idcok |
| idllok | immedok |
| in_wch | in_wchnstr |
| in_wchstr | inch |
| inchnstr | inchstr |
| init_color | init_pair |
| initscr | innstr |
| innwstr | ins_nwstr |
| ins_wch | ins_wstr |
| insch | insdelln |

libcurses(3XCURSES)

| | |
|----------------|---------------|
| insertln | insnstr |
| insstr | instr |
| intrflush | inwstr |
| is_linetouched | is_wintouched |
| isendwin | key_name |
| keyname | keypad |
| killchar | killwchar |
| leaveok | longname |
| meta | move |
| mvadd_wch | mvadd_wchnstr |
| mvadd_wchstr | mvaddch |
| mvaddchnstr | mvaddchstr |
| mvaddnstr | mvaddnwstr |
| mvaddstr | mvaddwstr |
| mvchgat | mvcur |
| mvdelch | mvderwin |
| mvget_wch | mvget_wstr |
| mvgetch | mvgetn_wstr |
| mvgetnstr | mvgetstr |
| mvhline | mvhline_set |
| mvin_wch | mvin_wchnstr |
| mvin_wchstr | mvinch |
| mvinchnstr | mvinchstr |
| mvinnstr | mvinnwstr |
| mvins_nwstr | mvins_wch |
| mvins_wstr | mvinsch |
| mvinsnstr | mvinsstr |
| mvinstr | mvinwstr |
| mvprintw | mvscanw |
| mvvline | mvvline_set |

| | |
|---------------|----------------|
| mvwadd_wch | mvwadd_wchnstr |
| mvwadd_wchstr | mvwaddch |
| mvwaddchnstr | mvwaddchstr |
| mvwaddnstr | mvwaddnwstr |
| mvwaddstr | mvwaddwstr |
| mvwchgat | mvwdelch |
| mvwget_wch | mvwget_wstr |
| mvwgetch | mvwgetn_wstr |
| mvwgetnstr | mvwgetstr |
| mvwhline | mvwhline_set |
| mvwin | mvwin_wch |
| mvwin_wchnstr | mvwin_wchstr |
| mvwinch | mvwinchnstr |
| mvwinchstr | mvwinnstr |
| mvwinnwstr | mvwins_nwstr |
| mvwins_wch | mvwins_wstr |
| mvwinsch | mvwinsnstr |
| mvwinsstr | mvwinstr |
| mvwinwstr | mvwprintw |
| mvwscanw | mvwvline |
| mvwvline_set | napms |
| newpad | newterm |
| newwin | nl |
| nocbreak | nodelay |
| noecho | nonl |
| noqiflush | noraw |
| notimeout | overlay |
| overwrite | pair_content |
| pecho_wchar | pechochar |
| pnoutrefresh | prefresh |

libcurses(3XCURSES)

| | |
|------------------|-----------------|
| printw | putp |
| putwin | qiflush |
| raw | redrawwin |
| refresh | reset_prog_mode |
| reset_shell_mode | resetty |
| restartterm | riporffline |
| savetty | scanw |
| scr_dump | scr_init |
| scr_restore | scr_set |
| scrl | scroll |
| scrollok | set_curterm |
| set_term | setcchar |
| setscreg | setupterm |
| slk_attr_off | slk_attr_on |
| slk_attr_set | slk_attroff |
| slk_attron | slk_attrset |
| slk_clear | slk_color |
| slk_init | slk_label |
| slk_noutrefresh | slk_refresh |
| slk_restore | slk_set |
| slk_touch | slk_wset |
| standend | standout |
| start_color | stdscr |
| subpad | subwin |
| syncok | term_attrs |
| termattrs | termname |
| tgetent | tgetflag |
| tgetnum | tgetstr |
| tgoto | tigetflag |
| tigetnum | tigetstr |

| | |
|--------------|-------------|
| timeout | touchline |
| touchwin | tparm |
| tputs | typeahead |
| unctrl | unget_wch |
| ungetch | untouchwin |
| use_env | vid_attr |
| vid_puts | vidattr |
| vidputs | vline |
| vline_set | vw_printw |
| vw_scanw | vwprintw |
| vwscanw | wadd_wch |
| wadd_wchnstr | wadd_wchstr |
| waddch | waddchnstr |
| waddchstr | waddnstr |
| waddnwstr | waddstr |
| waddwstr | wattr_get |
| wattr_off | wattr_on |
| wattr_set | wattroff |
| wattron | wattrset |
| wbkgd | wbkgdset |
| wbkgrnd | wbkgrndset |
| wborder | wborder_set |
| wchgat | wclear |
| wclrtoobot | wclrtoeol |
| wcolor_set | wcursyncup |
| wdelch | wdeleteln |
| wecho_wchar | wechochar |
| werase | wget_wch |
| wget_wstr | wgetbkgrnd |
| wgetch | wgetn_wstr |

libcurses(3XCURSES)

| | |
|------------|--------------|
| wgetnstr | wgetstr |
| whline | whline_set |
| win_wch | win_wchnstr |
| win_wchstr | winch |
| winchnstr | winchstr |
| winnstr | winnwstr |
| wins_nwstr | wins_wch |
| wins_wstr | winsch |
| winsdelln | winsertln |
| winsnstr | winsstr |
| winstr | winwstr |
| wmove | wnoutrefresh |
| wprintw | wredrawln |
| wrefresh | wscanw |
| wscr1 | wsetscrreg |
| wstandend | wstandout |
| wsyncdown | wsyncup |
| wtimeout | wtouchln |
| wunctrl | wvline |
| wvline_set | |

FILES /usr/xpg4/lib/libcurses.so.1
shared object for backward compatibility

/usr/xpg4/lib/libcurses.so.2
shared object

/usr/xpg4/lib/64/libcurses.so.1
64-bit shared object for backward compatibility

/usr/xpg4/lib/64/libcurses.so.2
64-bit shared object

NOTES The libcurses.so.1 listed above is an earlier shared object that provides the previous version of the X/Open Curses library (Issue 4). There is no binary compatibility between libcurses.so.1 and libcurses.so.2. This file is provided for backwards compatibility and will be removed in a future Solaris release. There is no plan to fix any of its defects.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|---------------------------------------|
| Availability | SUNWcsl (32-bit) SUNWcslx (64-bit) |
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `intro(3)`, `curses(3XCURSES)`, `libcurses(3LIB)`, `libcurses(3LIBUCB)`, `attributes(5)`, `standards(5)`

line(3PLOT)

| | |
|--------------------|--|
| NAME | plot, arc, box, circle, closepl, closevt, cont, erase, label, line, linemod, move, openpl, openvt, point, space – graphics interface |
| SYNOPSIS | <pre>cc [flag ...] file ... -lplot [library...] #include <plot.h> void arc(short x0, short y0, short x1, short y1, short x2, short y2); void box(short x0, short y0, short x1, short y1); void circle(short x, short y, short r); void closepl(); void closevt(); void cont(short x, short y); void erase(); void label(char *s); void line(short x0, short y0, short x1, short y1); void linemod(char *s); void move(short x, short y); void openpl(); void openvt(); void point(short x, short y); void space(short x0, short y0, short x1, short y1);</pre> |
| DESCRIPTION | <p>These functions generate graphics output for a set of output devices. The format of the output is dependent upon which link editor option is used when the program is compiled and linked (see Link Editor).</p> <p>The term "current point" refers to the current setting for the x and y coordinates.</p> <p>The <code>arc()</code> function specifies a circular arc. The coordinates $(x0, y0)$ specify the center of the arc. The coordinates $(x1, y1)$ specify the starting point of the arc. The coordinates $(x2, y2)$ specify the end point of the circular arc.</p> <p>The <code>box()</code> function specifies a rectangle with coordinates $(x0, y0)$, $(x0, y1)$, $(x1, y0)$, and $(x1, y1)$. The current point is set to $(x1, y1)$.</p> <p>The <code>circle()</code> function specifies a circle with a center at the coordinates (x, y) and a radius of r.</p> <p>The <code>closevt()</code> and <code>closepl()</code> functions flush the output.</p> <p>The <code>cont()</code> function specifies a line beginning at the current point and ending at the coordinates (x, y). The current point is set to (x, y).</p> |

The `erase()` function starts another frame of output.

The `label()` function places the null terminated string `s` so that the first character falls on the current point. The string is then terminated by a NEWLINE character.

The `line()` function draws a line starting at the coordinates $(x0, y0)$ and ending at the coordinates $(x1, y1)$. The current point is set to $(x1, y1)$.

The `linemod()` function specifies the style for drawing future lines. `s` may contain one of the following: `dotted`, `solid`, `longdashed`, `shortdashed`, or `dotdashed`.

The `move()` function sets the current point to the coordinates (x, y) .

The `openpl()` or `openvt()` function must be called to open the device before any other plot functions are called.

The `point()` function plots the point given by the coordinates (x, y) . The current point is set to (x, y) .

The `space()` function specifies the size of the plotting area. The plot will be reduced or enlarged as necessary to fit the area specified. The coordinates $(x0, y0)$ specify the lower left hand corner of the plotting area. The coordinates $(x1, y1)$ specify the upper right hand corner of the plotting area.

Link Editor Various flavors of these functions exist for different output devices. They are obtained by using the following `ld(1)` options:

```
-lplot          device-independent graphics stream on standard output in the
                 format described in plot(4B)
-l300           GSI 300 terminal
-l300s          GSI 300S terminal
-l4014          Tektronix 4014 terminal
-l450           GSI 450 terminal
-lvt0
```

FILES

```
/usr/lib/libplot.a
  archive library
/usr/lib/libplot.so.1
  shared object
/usr/lib/sparcv9/libplot.so.1
  64-bit shared object
/usr/lib/lib300.a
  archive library
/usr/lib/lib300.so.1
  shared object
```

line(3PLOT)

/usr/lib/sparcv9/lib300.so.1
64-bit shared object

/usr/lib/lib300s.a
archive library

/usr/lib/lib300s.so.1
shared object

/usr/lib/sparcv9/lib300s.so.1
64-bit shared object

/usr/lib/lib4014.a
archive library

/usr/lib/lib4014.so.1
shared object

/usr/lib/sparcv9/lib4014.so.1
64-bit shared object

/usr/lib/lib450.a
archive library

/usr/lib/lib450.so.1
shared object

/usr/lib/sparcv9/lib450.so.1
64-bit shared object

/usr/lib/libvt0.a
archive library

/usr/lib/libvt0.so.1
shared object

/usr/lib/sparcv9/libvt0.so.1
64-bit shared object

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `graph(1)`, `ld(1)`, `libplot(3LIB)`, `plot(4B)`, `attributes(5)`

| | |
|--------------------|--|
| NAME | plot, arc, box, circle, closepl, closevt, cont, erase, label, line, linemod, move, openpl, openvt, point, space – graphics interface |
| SYNOPSIS | <pre>cc [flag ...] file ... -lplot [library...] #include <plot.h> void arc(short x0, short y0, short x1, short y1, short x2, short y2); void box(short x0, short y0, short x1, short y1); void circle(short x, short y, short r); void closepl(); void closevt(); void cont(short x, short y); void erase(); void label(char *s); void line(short x0, short y0, short x1, short y1); void linemod(char *s); void move(short x, short y); void openpl(); void openvt(); void point(short x, short y); void space(short x0, short y0, short x1, short y1);</pre> |
| DESCRIPTION | <p>These functions generate graphics output for a set of output devices. The format of the output is dependent upon which link editor option is used when the program is compiled and linked (see Link Editor).</p> <p>The term "current point" refers to the current setting for the x and y coordinates.</p> <p>The <code>arc()</code> function specifies a circular arc. The coordinates $(x0, y0)$ specify the center of the arc. The coordinates $(x1, y1)$ specify the starting point of the arc. The coordinates $(x2, y2)$ specify the end point of the circular arc.</p> <p>The <code>box()</code> function specifies a rectangle with coordinates $(x0, y0)$, $(x0, y1)$, $(x1, y0)$, and $(x1, y1)$. The current point is set to $(x1, y1)$.</p> <p>The <code>circle()</code> function specifies a circle with a center at the coordinates (x, y) and a radius of r.</p> <p>The <code>closevt()</code> and <code>closepl()</code> functions flush the output.</p> <p>The <code>cont()</code> function specifies a line beginning at the current point and ending at the coordinates (x, y). The current point is set to (x, y).</p> |

linemod(3PLOT)

The `erase()` function starts another frame of output.

The `label()` function places the null terminated string `s` so that the first character falls on the current point. The string is then terminated by a NEWLINE character.

The `line()` function draws a line starting at the coordinates (x_0, y_0) and ending at the coordinates (x_1, y_1) . The current point is set to (x_1, y_1) .

The `linemod()` function specifies the style for drawing future lines. `s` may contain one of the following: `dotted`, `solid`, `longdashed`, `shortdashed`, or `dotdashed`.

The `move()` function sets the current point to the coordinates (x, y) .

The `openpl()` or `openvt()` function must be called to open the device before any other plot functions are called.

The `point()` function plots the point given by the coordinates (x, y) . The current point is set to (x, y) .

The `space()` function specifies the size of the plotting area. The plot will be reduced or enlarged as necessary to fit the area specified. The coordinates (x_0, y_0) specify the lower left hand corner of the plotting area. The coordinates (x_1, y_1) specify the upper right hand corner of the plotting area.

Link Editor Various flavors of these functions exist for different output devices. They are obtained by using the following `ld(1)` options:

| | |
|---------------------|--|
| <code>-lplot</code> | device-independent graphics stream on standard output in the format described in <code>plot(4B)</code> |
| <code>-l300</code> | GSI 300 terminal |
| <code>-l300s</code> | GSI 300S terminal |
| <code>-l4014</code> | Tektronix 4014 terminal |
| <code>-l450</code> | GSI 450 terminal |
| <code>-lvt0</code> | |

FILES

| | |
|--|----------------------|
| <code>/usr/lib/libplot.a</code> | archive library |
| <code>/usr/lib/libplot.so.1</code> | shared object |
| <code>/usr/lib/sparcv9/libplot.so.1</code> | 64-bit shared object |
| <code>/usr/lib/lib300.a</code> | archive library |
| <code>/usr/lib/lib300.so.1</code> | shared object |

```

/usr/lib/sparcv9/lib300.so.1
  64-bit shared object
/usr/lib/lib300s.a
  archive library
/usr/lib/lib300s.so.1
  shared object
/usr/lib/sparcv9/lib300s.so.1
  64-bit shared object
/usr/lib/lib4014.a
  archive library
/usr/lib/lib4014.so.1
  shared object
/usr/lib/sparcv9/lib4014.so.1
  64-bit shared object
/usr/lib/lib450.a
  archive library
/usr/lib/lib450.so.1
  shared object
/usr/lib/sparcv9/lib450.so.1
  64-bit shared object
/usr/lib/libvt0.a
  archive library
/usr/lib/libvt0.so.1
  shared object
/usr/lib/sparcv9/libvt0.so.1
  64-bit shared object

```

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `graph(1)`, `ld(1)`, `libplot(3LIB)`, `plot(4B)`, `attributes(5)`

LINES(3XCURSES)

- NAME** LINES – number of lines on terminal screen
- SYNOPSIS**

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \  
-R /usr/xpg4/lib -lcurses [ library... ]
```



```
c89 [ flag... ] file... -lcurses [ library ... ]
```



```
#include <curses.h>
```



```
extern int LINES;
```
- DESCRIPTION** The external variable LINES indicates the number of lines on the terminal screen.
- ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

- SEE ALSO** `initscr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | form_field_new, new_field, dup_field, link_field, free_field – create and destroy forms fields | | | | | | | | |
|----------------------|--|----------------|-------------------------------------|-------------|---|----------------|---------------|----------------|---------------------------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> FIELD *new_field(int r, int c, int frow, int fcol, int nrow, int ncol); FIELD *dup_field(FIELD *field, int frow, int fcol); FIELD *link_field(FIELD *field, int frow, int fcol); int free_field(FIELD *field);</pre> | | | | | | | | |
| DESCRIPTION | <p>new_field() creates a new field with <i>r</i> rows and <i>c</i> columns, starting at <i>frow</i>, <i>fcol</i>, in the subwindow of a form. <i>nrow</i> is the number of off-screen rows and <i>nbuf</i> is the number of additional working buffers. This routine returns a pointer to the new field.</p> <p>dup_field() duplicates <i>field</i> at the specified location. All field attributes are duplicated, including the current contents of the field buffers.</p> <p>link_field() also duplicates <i>field</i> at the specified location. However, unlike dup_field(), the new field shares the field buffers with the original field. After creation, the attributes of the new field can be changed without affecting the original field.</p> <p>free_field() frees the storage allocated for <i>field</i>.</p> | | | | | | | | |
| RETURN VALUES | <p>Routines that return pointers return NULL on error. free_field() returns one of the following:</p> <table border="0"> <tr> <td>E_OK</td> <td>The function returned successfully.</td> </tr> <tr> <td>E_CONNECTED</td> <td>The field is already connected to a form.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> <tr> <td>E_BAD_ARGUMENT</td> <td>An argument is incorrect.</td> </tr> </table> | E_OK | The function returned successfully. | E_CONNECTED | The field is already connected to a form. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An argument is incorrect. |
| E_OK | The function returned successfully. | | | | | | | | |
| E_CONNECTED | The field is already connected to a form. | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | |
| E_BAD_ARGUMENT | An argument is incorrect. | | | | | | | | |
| ATTRIBUTES | <p>See attributes(5) for descriptions of the following attributes:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | | | |
| MT-Level | Unsafe | | | | | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | | | | | |

link_fieldtype(3CURSES)

| | | | | | | | | | |
|----------------------|--|------|-------------------------------------|----------------|---------------|----------------|---------------------------|-------------|--|
| NAME | form_fieldtype, new_fieldtype, free_fieldtype, set_fieldtype_arg, set_fieldtype_choice, link_fieldtype – forms fieldtype routines | | | | | | | | |
| SYNOPSIS | <pre>cc [<i>flag...</i>] <i>file...</i> -lform -lcurses [<i>library...</i>] #include <form.h> FIELDTYPE *new_fieldtype(int (* <i>field_check</i>)(FIELD *, char *), int (*<i>char_check</i>)(int, char *)); int free_fieldtype(FIELDTYPE *<i>fieldtype</i>); int set_fieldtype_arg(FIELDTYPE *<i>fieldtype</i>, char *(* <i>mak_arg</i>)(va_list *), char *(* <i>copy_arg</i>)(char *), void (* <i>free_arg</i>)(char *)); int set_fieldtype_choice(FIELDTYPE *<i>fieldtype</i>, int (* <i>next_choice</i>)(FIELD *, char *), int (*<i>prev_choice</i>)(FIELD *, char *)); FIELDTYPE *link_fieldtype(FIELDTYPE *<i>type1</i>, FIELDTYPE *<i>type2</i>);</pre> | | | | | | | | |
| DESCRIPTION | <p>new_fieldtype() creates a new field type. The application programmer must write the function <i>field_check</i>, which validates the field value, and the function <i>char_check</i>, which validates each character. free_fieldtype() frees the space allocated for the field type.</p> <p>By associating function pointers with a field type, set_fieldtype_arg() connects to the field type additional arguments necessary for a set_field_type() call. Function <i>mak_arg</i> allocates a structure for the field specific parameters to set_field_type() and returns a pointer to the saved data. Function <i>copy_arg</i> duplicates the structure created by <i>make_arg</i>. Function <i>free_arg</i> frees any storage allocated by <i>make_arg</i> or <i>copy_arg</i>.</p> <p>The form_driver() requests REQ_NEXT_CHOICE and REQ_PREV_CHOICE let the user request the next or previous value of a field type comprising an ordered set of values. set_fieldtype_choice() allows the application programmer to implement these requests for the given field type. It associates with the given field type those application-defined functions that return pointers to the next or previous choice for the field.</p> <p>link_fieldtype() returns a pointer to the field type built from the two given types. The constituent types may be any application-defined or pre-defined types.</p> | | | | | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <table><tr><td>E_OK</td><td>The function returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr><tr><td>E_BAD_ARGUMENT</td><td>An argument is incorrect.</td></tr><tr><td>E_CONNECTED</td><td>Type is connected to one or more fields.</td></tr></table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An argument is incorrect. | E_CONNECTED | Type is connected to one or more fields. |
| E_OK | The function returned successfully. | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | |
| E_BAD_ARGUMENT | An argument is incorrect. | | | | | | | | |
| E_CONNECTED | Type is connected to one or more fields. | | | | | | | | |

link_fieldtype(3CURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3CURSES)`, `forms(3CURSES)`, `attributes(5)`

NOTES The header `<form.h>` automatically includes the headers `<eti.h>` and `< curses.h>`.

longname(3CURSES)

| | |
|----------------------|---|
| NAME | <code>curs_termattrs</code> , <code>baudrate</code> , <code>erasechar</code> , <code>has_ic</code> , <code>has_il</code> , <code>killchar</code> , <code>longname</code> , <code>termattrs</code> , <code>termname</code> – curses environment query routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int baudrate(void); char erasechar(void); int has_ic(void); int has_il(void); char killchar(void); char *longname(void); chtype termattrs(void); char *termname(void);</pre> |
| DESCRIPTION | <p>The <code>baudrate()</code> routine returns the output speed of the terminal. The number returned is in bits per second, for example 9600, and is an integer.</p> <p>With the <code>erasechar()</code> routine, the user's current erase character is returned.</p> <p>The <code>has_ic()</code> routine is true if the terminal has insert- and delete-character capabilities.</p> <p>The <code>has_il()</code> routine is true if the terminal has insert- and delete-line capabilities, or can simulate them using scrolling regions. This might be used to determine if it would be appropriate to turn on physical scrolling using <code>scrollok()</code>.</p> <p>With the <code>killchar()</code> routine, the user's current line kill character is returned.</p> <p>The <code>longname()</code> routine returns a pointer to a static area containing a verbose description of the current terminal. The maximum length of a verbose description is 128 characters. It is defined only after the call to <code>initscr()</code> or <code>newterm()</code>. The area is overwritten by each call to <code>newterm()</code> and is not restored by <code>set_term()</code>, so the value should be saved between calls to <code>newterm()</code> if <code>longname()</code> is going to be used with multiple terminals.</p> <p>If a given terminal doesn't support a video attribute that an application program is trying to use, <code>curses</code> may substitute a different video attribute for it. The <code>termattrs()</code> function returns a logical OR of all video attributes supported by the terminal. This information is useful when a <code>curses</code> program needs complete control over the appearance of the screen.</p> <p>The <code>termname()</code> routine returns the value of the environment variable <code>TERM</code> (truncated to 14 characters).</p> |
| RETURN VALUES | <code>longname()</code> and <code>termname()</code> return <code>NULL</code> on error. |

longname(3CURSES)

Routines that return an integer return ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_initscr(3CURSES)`, `curl_outopts(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `termattrs()` may be a macro.

longname(3XCURSES)

NAME | longname – return full terminal type name

SYNOPSIS |

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

const char *longname(void);
```

DESCRIPTION | The longname() function returns a pointer to a static area containing a verbose description (128 characters or fewer) of the terminal. The area is defined after calls to initscr(3XCURSES), newterm(3XCURSES), or setupterm(3XCURSES). The value should be saved if longname() is going to be used with multiple terminals since it will be overwritten with a new value after each call to newterm() or setupterm().

RETURN VALUES | On success, the longname() function returns a pointer to a verbose description of the terminal. Otherwise, it returns a null pointer.

ERRORS | None.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | initscr(3XCURSES), libcurses(3XCURSES), newterm(3XCURSES), setupterm(3XCURSES), attributes(5), standards(5)

NAME menu_attributes, set_menu_fore, menu_fore, set_menu_back, menu_back, set_menu_grey, menu_grey, set_menu_pad, menu_pad – control menus display attributes

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

int set_menu_fore(MENU *menu, chtype attr);
chtype menu_fore(MENU *menu);

int set_menu_back(MENU *menu, chtype attr);
chtype menu_back(MENU *menu);

int set_menu_grey(MENU *menu, chtype attr);
chtype menu_grey(MENU *menu);

int set_menu_pad(MENU *menu, int pad);
int menu_pad(MENU *menu);
```

DESCRIPTION set_menu_fore() sets the foreground attribute of *menu* — the display attribute for the current item (if selectable) on single-valued menus and for selected items on multi-valued menus. This display attribute is a curses library visual attribute. menu_fore() returns the foreground attribute of *menu*.

set_menu_back() sets the background attribute of *menu* — the display attribute for unselected, yet selectable, items. This display attribute is a curses library visual attribute.

set_menu_grey() sets the grey attribute of *menu* — the display attribute for nonselectable items in multi-valued menus. This display attribute is a curses library visual attribute. menu_grey() returns the grey attribute of *menu*.

The pad character is the character that fills the space between the name and description of an item. set_menu_pad() sets the pad character for *menu* to *pad*. menu_pad() returns the pad character of *menu*.

RETURN VALUES These routines return one of the following:

- E_OK The routine returned successfully.
- E_SYSTEM_ERROR System error.
- E_BAD_ARGUMENT An incorrect argument was passed to the routine.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

menu_attributes(3CURSES)

SEE ALSO | curses(3CURSES), menus(3CURSES), attributes(5)

NOTES | The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

menu_back(3CURSES)

NAME menu_attributes, set_menu_fore, menu_fore, set_menu_back, menu_back, set_menu_grey, menu_grey, set_menu_pad, menu_pad – control menus display attributes

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

int set_menu_fore(MENU *menu, chtype attr);
chtype menu_fore(MENU *menu);

int set_menu_back(MENU *menu, chtype attr);
chtype menu_back(MENU *menu);

int set_menu_grey(MENU *menu, chtype attr);
chtype menu_grey(MENU *menu);

int set_menu_pad(MENU *menu, int pad);
int menu_pad(MENU *menu);
```

DESCRIPTION set_menu_fore() sets the foreground attribute of *menu* — the display attribute for the current item (if selectable) on single-valued menus and for selected items on multi-valued menus. This display attribute is a curses library visual attribute. menu_fore() returns the foreground attribute of *menu*.

set_menu_back() sets the background attribute of *menu* — the display attribute for unselected, yet selectable, items. This display attribute is a curses library visual attribute.

set_menu_grey() sets the grey attribute of *menu* — the display attribute for nonselectable items in multi-valued menus. This display attribute is a curses library visual attribute. menu_grey() returns the grey attribute of *menu*.

The pad character is the character that fills the space between the name and description of an item. set_menu_pad() sets the pad character for *menu* to *pad*. menu_pad() returns the pad character of *menu*.

RETURN VALUES These routines return one of the following:

| | |
|----------------|--|
| E_OK | The routine returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

menu_back(3CURSES)

SEE ALSO | curses(3CURSES), menus(3CURSES), attributes(5)

NOTES | The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

NAME menu_cursor, pos_menu_cursor – correctly position a menus cursor

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

int pos_menu_cursor(MENU *menu);
```

DESCRIPTION pos_menu_cursor() moves the cursor in the window of *menu* to the correct position to resume menu processing. This is needed after the application calls a curses library I/O routine.

RETURN VALUES This routine returns one of the following:

E_OK The routine returned successfully.

E_SYSTEM_ERROR System error.

E_BAD_ARGUMENT An incorrect argument was passed to the routine.

E_NOT_POSTED The menu has not been posted.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), panel_update(3CURSES), panels(3CURSES), attributes(5)

NOTES The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

menu_driver(3CURSES)

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----------------------|--|---------------|------------------------------------|----------------|------------------------|----------------|--|---------------|-----------------------|---------------|-------------------|---------------|---------------------|---------------|-------------------|---------------|---------------------|----------------|-------------------------|---------------|------------------------|---------------|------------------------|---------------|----------------------------|-----------------|---------------------------|-------------------|--------------------------------|------------------|--|----------------|------------------------------|----------------|-------------------------------------|
| NAME | menu_driver – command processor for the menus subsystem | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int menu_driver(MENU *menu, int c);</pre> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DESCRIPTION | <p>menu_driver() is the workhorse of the menus subsystem. It checks to determine whether the character <i>c</i> is a menu request or data. If <i>c</i> is a request, the menu driver executes the request and reports the result. If <i>c</i> is data (a printable ASCII character), it enters the data into the pattern buffer and tries to find a matching item. If no match is found, the menu driver deletes the character from the pattern buffer and returns E_NO_MATCH. If the character is not recognized, the menu driver assumes it is an application-defined command and returns E_UNKNOWN_COMMAND.</p> <p>Menu driver requests:</p> <table><tr><td>REQ_LEFT_ITEM</td><td>Move left to an item.</td></tr><tr><td>REQ_RIGHT_ITEM</td><td>Move right to an item.</td></tr><tr><td>REQ_UP_ITEM</td><td>Move up to an item.</td></tr><tr><td>REQ_DOWN_ITEM</td><td>Move down to an item.</td></tr><tr><td>REQ_SCR_ULINE</td><td>Scroll up a line.</td></tr><tr><td>REQ_SCR_DLINE</td><td>Scroll down a line.</td></tr><tr><td>REQ_SCR_DPAGE</td><td>Scroll up a page.</td></tr><tr><td>REQ_SCR_UPAGE</td><td>Scroll down a page.</td></tr><tr><td>REQ_FIRST_ITEM</td><td>Move to the first item.</td></tr><tr><td>REQ_LAST_ITEM</td><td>Move to the last item.</td></tr><tr><td>REQ_NEXT_ITEM</td><td>Move to the next item.</td></tr><tr><td>REQ_PREV_ITEM</td><td>Move to the previous item.</td></tr><tr><td>REQ_TOGGLE_ITEM</td><td>Select/de-select an item.</td></tr><tr><td>REQ_CLEAR_PATTERN</td><td>Clear the menu pattern buffer.</td></tr><tr><td>REQ_BACK_PATTERN</td><td>Delete the previous character from pattern buffer.</td></tr><tr><td>REQ_NEXT_MATCH</td><td>Move the next matching item.</td></tr><tr><td>REQ_PREV_MATCH</td><td>Move to the previous matching item.</td></tr></table> | REQ_LEFT_ITEM | Move left to an item. | REQ_RIGHT_ITEM | Move right to an item. | REQ_UP_ITEM | Move up to an item. | REQ_DOWN_ITEM | Move down to an item. | REQ_SCR_ULINE | Scroll up a line. | REQ_SCR_DLINE | Scroll down a line. | REQ_SCR_DPAGE | Scroll up a page. | REQ_SCR_UPAGE | Scroll down a page. | REQ_FIRST_ITEM | Move to the first item. | REQ_LAST_ITEM | Move to the last item. | REQ_NEXT_ITEM | Move to the next item. | REQ_PREV_ITEM | Move to the previous item. | REQ_TOGGLE_ITEM | Select/de-select an item. | REQ_CLEAR_PATTERN | Clear the menu pattern buffer. | REQ_BACK_PATTERN | Delete the previous character from pattern buffer. | REQ_NEXT_MATCH | Move the next matching item. | REQ_PREV_MATCH | Move to the previous matching item. |
| REQ_LEFT_ITEM | Move left to an item. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_RIGHT_ITEM | Move right to an item. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_UP_ITEM | Move up to an item. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_DOWN_ITEM | Move down to an item. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_SCR_ULINE | Scroll up a line. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_SCR_DLINE | Scroll down a line. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_SCR_DPAGE | Scroll up a page. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_SCR_UPAGE | Scroll down a page. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_FIRST_ITEM | Move to the first item. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_LAST_ITEM | Move to the last item. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_NEXT_ITEM | Move to the next item. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_PREV_ITEM | Move to the previous item. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_TOGGLE_ITEM | Select/de-select an item. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_CLEAR_PATTERN | Clear the menu pattern buffer. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_BACK_PATTERN | Delete the previous character from pattern buffer. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_NEXT_MATCH | Move the next matching item. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| REQ_PREV_MATCH | Move to the previous matching item. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| RETURN VALUES | <p>menu_driver() returns one of the following:</p> <table><tr><td>E_OK</td><td>The routine returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr><tr><td>E_BAD_ARGUMENT</td><td>An incorrect argument was passed to the routine.</td></tr></table> | E_OK | The routine returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An incorrect argument was passed to the routine. | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| E_OK | The routine returned successfully. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

menu_driver(3CURSES)

E_BAD_STATE The routine was called from an initialization or termination function.

E_NOT_POSTED The menu has not been posted.

E_UNKNOWN_COMMAND An unknown request was passed to the menu driver.

E_NO_MATCH The character failed to match.

E_NOT_SELECTABLE The item cannot be selected.

E_REQUEST_DENIED The menu driver could not process the request.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3CURSES)`, `menus(3CURSES)`, `attributes(5)`

NOTES Application-defined commands should be defined relative to (greater than) `MAX_COMMAND`, the maximum value of a request listed above.

The header `<menu.h>` automatically includes the headers `<eti.h>` and `<curses.h>`.

menu_fore(3CURSES)

NAME menu_attributes, set_menu_fore, menu_fore, set_menu_back, menu_back, set_menu_grey, menu_grey, set_menu_pad, menu_pad – control menus display attributes

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

int set_menu_fore(MENU *menu, chtype attr);
chtype menu_fore(MENU *menu);

int set_menu_back(MENU *menu, chtype attr);
chtype menu_back(MENU *menu);

int set_menu_grey(MENU *menu, chtype attr);
chtype menu_grey(MENU *menu);

int set_menu_pad(MENU *menu, int pad);
int menu_pad(MENU *menu);
```

DESCRIPTION

set_menu_fore() sets the foreground attribute of *menu* — the display attribute for the current item (if selectable) on single-valued menus and for selected items on multi-valued menus. This display attribute is a curses library visual attribute. menu_fore() returns the foreground attribute of *menu*.

set_menu_back() sets the background attribute of *menu* — the display attribute for unselected, yet selectable, items. This display attribute is a curses library visual attribute.

set_menu_grey() sets the grey attribute of *menu* — the display attribute for nonselectable items in multi-valued menus. This display attribute is a curses library visual attribute. menu_grey() returns the grey attribute of *menu*.

The pad character is the character that fills the space between the name and description of an item. set_menu_pad() sets the pad character for *menu* to *pad*. menu_pad() returns the pad character of *menu*.

RETURN VALUES These routines return one of the following:

| | |
|----------------|--|
| E_OK | The routine returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

menu_fore(3CURSES)

SEE ALSO | curses(3CURSES), menus(3CURSES), attributes(5)

NOTES | The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

menu_format(3CURSES)

| NAME | menu_format, set_menu_format – set and get maximum numbers of rows and columns in menus | | | | | | | | |
|----------------------|---|----------------|------------------------------------|----------------|---------------|----------------|--|----------|-----------------------------|
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int set_menu_format(MENU *menu, int rows, int cols); void menu_format(MENU *menu, int *rows, int *cols);</pre> | | | | | | | | |
| DESCRIPTION | <p>set_menu_format() sets the maximum number of rows and columns of items that may be displayed at one time on a menu. If the menu contains more items than can be displayed at once, the menu will be scrollable.</p> <p>menu_format() returns the maximum number of rows and columns that may be displayed at one time on <i>menu</i>. <i>rows</i> and <i>cols</i> are pointers to the variables used to return these values.</p> | | | | | | | | |
| RETURN VALUES | <p>set_menu_format() returns one of the following:</p> <table><tr><td>E_OK</td><td>The routine returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr><tr><td>E_BAD_ARGUMENT</td><td>An incorrect argument was passed to the routine.</td></tr><tr><td>E_POSTED</td><td>The menu is already posted.</td></tr></table> | E_OK | The routine returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An incorrect argument was passed to the routine. | E_POSTED | The menu is already posted. |
| E_OK | The routine returned successfully. | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. | | | | | | | | |
| E_POSTED | The menu is already posted. | | | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | | | |
| MT-Level | Unsafe | | | | | | | | |
| SEE ALSO | curses(3CURSES), menus(3CURSES), attributes(5) | | | | | | | | |
| NOTES | The header <menu.h> automatically includes the headers <eti.h> and <curses.h>. | | | | | | | | |

NAME menu_attributes, set_menu_fore, menu_fore, set_menu_back, menu_back, set_menu_grey, menu_grey, set_menu_pad, menu_pad – control menus display attributes

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

int set_menu_fore(MENU *menu, chtype attr);
chtype menu_fore(MENU *menu);

int set_menu_back(MENU *menu, chtype attr);
chtype menu_back(MENU *menu);

int set_menu_grey(MENU*menu, chtype attr);
chtype menu_grey(MENU *menu);

int set_menu_pad(MENU *menu, int pad);
int menu_pad(MENU *menu);
```

DESCRIPTION

set_menu_fore() sets the foreground attribute of *menu* — the display attribute for the current item (if selectable) on single-valued menus and for selected items on multi-valued menus. This display attribute is a curses library visual attribute. menu_fore() returns the foreground attribute of *menu*.

set_menu_back() sets the background attribute of *menu* — the display attribute for unselected, yet selectable, items. This display attribute is a curses library visual attribute.

set_menu_grey() sets the grey attribute of *menu* — the display attribute for nonselectable items in multi-valued menus. This display attribute is a curses library visual attribute. menu_grey() returns the grey attribute of *menu*.

The pad character is the character that fills the space between the name and description of an item. set_menu_pad() sets the pad character for *menu* to *pad*. menu_pad() returns the pad character of *menu*.

RETURN VALUES These routines return one of the following:

| | |
|----------------|--|
| E_OK | The routine returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

menu_grey(3CURSES)

SEE ALSO | curses(3CURSES), menus(3CURSES), attributes(5)

NOTES | The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

| | | | | | |
|----------------------|--|------|------------------------------------|----------------|---------------|
| NAME | menu_hook, set_item_init, item_init, set_item_term, item_term, set_menu_init, menu_init, set_menu_term, menu_term – assign application-specific routines for automatic invocation by menus | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int set_item_init(MENU *menu, void (*func)(MENU *)); int set_item_term(MENU *menu, void (*func)(MENU *)); void item_term(MENU *menu); int set_menu_init(MENU *menu, void (*func)(MENU *)); void menu_init(MENU *menu); int set_menu_term(MENU *menu, void (*func)(MENU *)); void menu_term(MENU *menu);</pre> | | | | |
| DESCRIPTION | <p>set_item_init() assigns the application-defined function to be called when the <i>menu</i> is posted and just after the current item changes. item_init() returns a pointer to the item initialization routine, if any, called when the <i>menu</i> is posted and just after the current item changes.</p> <p>set_item_term() assigns an application-defined function to be called when the <i>menu</i> is unposted and just before the current item changes. item_term() returns a pointer to the termination function, if any, called when the <i>menu</i> is unposted and just before the current item changes.</p> <p>set_menu_init() assigns an application-defined function to be called when the <i>menu</i> is posted and just after the top row changes on a posted menu. menu_init() returns a pointer to the menu initialization routine, if any, called when the <i>menu</i> is posted and just after the top row changes on a posted menu.</p> <p>set_menu_term() assigns an application-defined function to be called when the <i>menu</i> is unposted and just before the top row changes on a posted menu. menu_term() returns a pointer to the menu termination routine, if any, called when the <i>menu</i> is unposted and just before the top row changes on a posted menu.</p> | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <table border="0"> <tr> <td style="padding-right: 20px;">E_OK</td> <td>The routine returned successfully.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> </table> | E_OK | The routine returned successfully. | E_SYSTEM_ERROR | System error. |
| E_OK | The routine returned successfully. | | | | |
| E_SYSTEM_ERROR | System error. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |

menu_hook(3CURSES)

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), attributes(5)

NOTES The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

| | | | | | |
|----------------------|--|------|------------------------------------|----------------|---------------|
| NAME | menu_hook, set_item_init, item_init, set_item_term, item_term, set_menu_init, menu_init, set_menu_term, menu_term – assign application-specific routines for automatic invocation by menus | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int set_item_init(MENU *menu, void (*func)(MENU *)); int set_item_term(MENU *menu, void (*func)(MENU *)); void item_term(MENU *menu); int set_menu_init(MENU *menu, void (*func)(MENU *)); void menu_init(MENU *menu); int set_menu_term(MENU *menu, void (*func)(MENU *)); void menu_term(MENU *menu);</pre> | | | | |
| DESCRIPTION | <p>set_item_init() assigns the application-defined function to be called when the <i>menu</i> is posted and just after the current item changes. item_init() returns a pointer to the item initialization routine, if any, called when the <i>menu</i> is posted and just after the current item changes.</p> <p>set_item_term() assigns an application-defined function to be called when the <i>menu</i> is unposted and just before the current item changes. item_term() returns a pointer to the termination function, if any, called when the <i>menu</i> is unposted and just before the current item changes.</p> <p>set_menu_init() assigns an application-defined function to be called when the <i>menu</i> is posted and just after the top row changes on a posted menu. menu_init() returns a pointer to the menu initialization routine, if any, called when the <i>menu</i> is posted and just after the top row changes on a posted menu.</p> <p>set_menu_term() assigns an application-defined function to be called when the <i>menu</i> is unposted and just before the top row changes on a posted menu. menu_term() returns a pointer to the menu termination routine, if any, called when the <i>menu</i> is unposted and just before the top row changes on a posted menu.</p> | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <table border="0"> <tr> <td style="padding-right: 20px;">E_OK</td> <td>The routine returned successfully.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> </table> | E_OK | The routine returned successfully. | E_SYSTEM_ERROR | System error. |
| E_OK | The routine returned successfully. | | | | |
| E_SYSTEM_ERROR | System error. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |

menu_init(3CURSES)

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), attributes(5)

NOTES The header <menu.h> automatically includes the headers <eti.h> and < curses.h>.

NAME menu_item_current, set_current_item, current_item, set_top_row, top_row, item_index
 – set and get current menus items

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

int set_current_item(MENU *menu, ITEM *item);
ITEM *current_item(MENU *menu);
int set_top_row(MENU *menu, int row);
int top_row(MENU *menu);
int item_index(ITEM *item);
```

DESCRIPTION

The current item of a menu is the item where the cursor is currently positioned. `set_current_item()` sets the current item of *menu* to *item*. `current_item()` returns a pointer to the the current item in *menu*.

`set_top_row()` sets the top row of *menu* to *row*. The left-most item on the new top row becomes the current item. `top_row()` returns the number of the menu row currently displayed at the top of *menu*.

`item_index()` returns the index to the *item* in the item pointer array. The value of this index ranges from 0 through *N*-1, where *N* is the total number of items connected to the menu.

RETURN VALUES

`current_item()` returns NULL on error.

`top_row()` and `index_item()` return -1 on error.

`set_current_item()` and `set_top_row()` return one of the following:

| | |
|-----------------|--|
| E_OK | The routine returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. |
| E_BAD_STATE | The routine was called from an initialization or termination function. |
| E_NOT_CONNECTED | No items are connected to the menu. |

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3CURSES)`, `menus(3CURSES)`, `attributes(5)`

menu_item_current(3CURSES)

NOTES | The header `<menu.h>` automatically includes the headers `<eti.h>` and `<curses.h>`.

menu_item_name(3CURSES)

NAME menu_item_name, item_name, item_description – get menus item name and description

SYNOPSIS

```
cc [ flag ... ] file ... -lmenu -lcurses [ library .. ]
#include <menu.h>

char *item_name (ITEM *item) ;
char *item_description (ITEM *item) ;
```

DESCRIPTION item_name() returns a pointer to the name of *item*.
item_description() returns a pointer to the description of *item*.

RETURN VALUES These routines return NULL on error.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), menu_new(3CURSES), attributes(5)

NOTES The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

menu_item_new(3CURSES)

NAME menu_item_new, new_item, free_item – create and destroy menus items

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

ITEM *new_item(char *name, char *desc);
int free_item(ITEM *item);
```

DESCRIPTION new_item() creates a new item from *name* and *description*, and returns a pointer to the new item.

free_item() frees the storage allocated for *item*. Once an item is freed, the user can no longer connect it to a menu.

RETURN VALUES new_item() returns NULL on error.

free_item() returns one of the following:

| | |
|----------------|--|
| E_OK | The routine returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. |
| E_CONNECTED | One or more items are already connected to another menu. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), attributes(5)

NOTES The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

NAME menu_item_opts, set_item_opts, item_opts_on, item_opts_off, item_opts – menus item option routines

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

int set_item_opts (ITEM *item, OPTIONS opts);
int item_opts_on (ITEM *item, OPTIONS opts);
int item_opts_off (ITEM *item, OPTIONS opts);
OPTIONS item_opts (ITEM *item);
```

DESCRIPTION

set_item_opts() turns on the named options for *item* and turns off all other options. Options are boolean values that can be OR-ed together.

item_opts_on() turns on the named options for *item*; no other option is changed.

item_opts_off() turns off the named options for *item*; no other option is changed.

item_opts() returns the current options of *item*.

O_SELECTABLE The item can be selected during menu processing.

RETURN VALUES Except for item_opts(), these routines return one of the following:

E_OK The routine returned successfully.

E_SYSTEM_ERROR System error.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), attributes(5)

NOTES The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

menu_items(3CURSES)

| NAME | menu_items, set_menu_items, item_count – connect and disconnect items to and from menus | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int set_menu_items(MENU *menu, ITEM **items); ITEM **menu_items(MENU *menu); int item_count(MENU *menu);</pre> | | | | |
| DESCRIPTION | set_menu_items() changes the item pointer array connected to <i>menu</i> to the item pointer array <i>items</i> . menu_items() returns a pointer to the item pointer array connected to <i>menu</i> . item_count() returns the number of items in <i>menu</i> . | | | | |
| RETURN VALUES | menu_items() returns NULL on error. item_count() returns -1 on error. set_menu_items() returns one of the following: E_OK The routine returned successfully. E_SYSTEM_ERROR System error. E_BAD_ARGUMENT An incorrect argument was passed to the routine. E_POSTED The menu is already posted. E_CONNECTED One or more items are already connected to another menu. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), menus(3CURSES), attributes(5) | | | | |
| NOTES | The header <menu.h> automatically includes the headers <eti.h> and <curses.h>. | | | | |

menu_item_value(3CURSES)

NAME menu_item_value, set_item_value, item_value – set and get menus item values

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

int set_item_value (ITEM *item, int bool);
int item_value (ITEM *item);
```

DESCRIPTION Unlike single-valued menus, multi-valued menus enable the end-user to select one or more items from a menu. `set_item_value()` sets the selected value of the *item* — TRUE (selected) or FALSE (not selected). `set_item_value()` may be used only with multi-valued menus. To make a menu multi-valued, use `set_menu_opts` or `menu_opts_off()` to turn off the option `O_ONEVALUE`. (See `menu_opts(3CURSES)`).

`item_value()` returns the select value of *item*, either TRUE (selected) or FALSE (unselected).

RETURN VALUES `set_item_value()` returns one of the following:

| | |
|-------------------------------|--|
| <code>E_OK</code> | The routine returned successfully. |
| <code>E_SYSTEM_ERROR</code> | System error. |
| <code>E_REQUEST_DENIED</code> | The menu driver could not process the request. |

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3CURSES)`, `menus(3CURSES)`, `menu_opts(3CURSES)`, `attributes(5)`

NOTES The header `<menu.h>` automatically includes the headers `<eti.h>` and `<curses.h>`.

menu_item_visible(3CURSES)

NAME menu_item_visible, item_visible – tell if menu item is visible

SYNOPSIS

```
cc [ flag ... ] file ... -lmenu -lcurses [ library .. ]
#include <menu.h>

int item_visible (ITEM *item);
```

DESCRIPTION A menu item is visible if it currently appears in the subwindow of a posted menu. `item_visible()` returns TRUE if *item* is visible, otherwise it returns FALSE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), menu_new(3CURSES), attributes(5)

NOTES The header `<menu.h>` automatically includes the headers `<eti.h>` and `<curses.h>`.

menu_mark(3CURSES)

NAME menu_mark, set_menu_mark – menus mark string routines

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

int set_menu_mark(MENU *menu, char *mark);
char *menu_mark(MENU *menu);
```

DESCRIPTION menus displays mark strings to distinguish selected items in a menu (or the current item in a single-valued menu). set_menu_mark() sets the mark string of menu to mark. menu_mark() returns a pointer to the mark string of menu.

RETURN VALUES menu_mark() returns NULL on error. set_menu_mark() returns one of the following:

E_OK The routine returned successfully.

E_SYSTEM_ERROR System error.

E_BAD_ARGUMENT An incorrect argument was passed to the routine.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), attributes(5)

NOTES The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

NAME menu_new, new_menu, free_menu – create and destroy menus

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

MENU *new_menu (ITEM **items) ;
int free_menu (MENU *menu) ;
```

DESCRIPTION new_menu () creates a new menu connected to the item pointer array *items* and returns a pointer to the new menu.

free_menu () disconnects *menu* from its associated item pointer array and frees the storage allocated for the menu.

RETURN VALUES new_menu () returns NULL on error.

free_menu () returns one of the following:

E_OK The routine returned successfully.

E_SYSTEM_ERROR System error.

E_BAD_ARGUMENT An incorrect argument was passed to the routine.

E_POSTED The menu is already posted.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), attributes(5)

NOTES The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

menu_opts(3CURSES)

NAME menu_opts, set_menu_opts, menu_opts_on, menu_opts_off – menus option routines

SYNOPSIS `cc [flag...] file... -lmenu -lcurses [library...]
#include <menu.h>`

`OPTIONS menu_opts (MENU *menu) ;`
`int set_menu_opts (MENU *menu, OPTIONS opts) ;`
`int menu_opts_on (MENU *menu, OPTIONS opts) ;`
`int menu_opts_off (MENU *menu, OPTIONS opts) ;`

Menu Options `set_menu_opts ()` turns on the named options for *menu* and turns off all other options. Options are boolean values that can be OR-ed together.

`menu_opts_on ()` turns on the named options for *menu*; no other option is changed.

`menu_opts_off ()` turns off the named options for *menu*; no other option is changed.

`menu_opts ()` returns the current options of *menu*.

The following values can be OR'd together to create *opts*.

| | |
|---------------------------|--|
| <code>O_ONEVALUE</code> | Only one item can be selected from the menu. |
| <code>O_SHOWDESC</code> | Display the description of the items. |
| <code>O_ROWMAJOR</code> | Display the menu in row major order. |
| <code>O_IGNORECASE</code> | Ignore the case when pattern matching. |
| <code>O_SHOWMATCH</code> | Place the cursor within the item name when pattern matching. |
| <code>O_NONCYCLIC</code> | Make certain menu driver requests non-cyclic. |

RETURN VALUES Except for `menu_opts ()`, these routines return one of the following:

| | |
|-----------------------------|------------------------------------|
| <code>E_OK</code> | The routine returned successfully. |
| <code>E_SYSTEM_ERROR</code> | System error. |
| <code>E_POSTED</code> | The menu is already posted. |

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3CURSES)`, `menus(3CURSES)`, `attributes(5)`

NOTES The header `<menu.h>` automatically includes the headers `<eti.h>` and `<curses.h>`.

NAME menu_opts, set_menu_opts, menu_opts_on, menu_opts_off – menus option routines

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

OPTIONS menu_opts (MENU *menu) ;
int set_menu_opts (MENU *menu, OPTIONS opts) ;
int menu_opts_on (MENU *menu, OPTIONS opts) ;
int menu_opts_off (MENU *menu, OPTIONS opts) ;
```

Menu Options

set_menu_opts () turns on the named options for *menu* and turns off all other options. Options are boolean values that can be OR-ed together.

menu_opts_on () turns on the named options for *menu*; no other option is changed.

menu_opts_off () turns off the named options for *menu*; no other option is changed.

menu_opts () returns the current options of *menu*.

The following values can be OR'd together to create *opts*.

| | |
|--------------|--|
| O_ONEVALUE | Only one item can be selected from the menu. |
| O_SHOWDESC | Display the description of the items. |
| O_ROWMAJOR | Display the menu in row major order. |
| O_IGNORECASE | Ignore the case when pattern matching. |
| O_SHOWMATCH | Place the cursor within the item name when pattern matching. |
| O_NONCYCLIC | Make certain menu driver requests non-cyclic. |

RETURN VALUES Except for menu_opts (), these routines return one of the following:

| | |
|----------------|------------------------------------|
| E_OK | The routine returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_POSTED | The menu is already posted. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), attributes(5)

NOTES The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

menu_opts_on(3CURSES)

| NAME | menu_opts, set_menu_opts, menu_opts_on, menu_opts_off – menus option routines | | | | | | | | | | | | |
|----------------------|--|----------------|--|----------------|---------------------------------------|------------|--------------------------------------|--------------|--|-------------|--|-------------|---|
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> OPTIONS menu_opts (MENU *menu) ; int set_menu_opts (MENU *menu, OPTIONS opts) ; int menu_opts_on (MENU *menu, OPTIONS opts) ; int menu_opts_off (MENU *menu, OPTIONS opts) ;</pre> | | | | | | | | | | | | |
| Menu Options | <p>set_menu_opts () turns on the named options for <i>menu</i> and turns off all other options. Options are boolean values that can be OR-ed together.</p> <p>menu_opts_on () turns on the named options for <i>menu</i>; no other option is changed.</p> <p>menu_opts_off () turns off the named options for <i>menu</i>; no other option is changed.</p> <p>menu_opts () returns the current options of <i>menu</i>.</p> <p>The following values can be OR'd together to create <i>opts</i>.</p> <table border="0"> <tr> <td>O_ONEVALUE</td> <td>Only one item can be selected from the menu.</td> </tr> <tr> <td>O_SHOWDESC</td> <td>Display the description of the items.</td> </tr> <tr> <td>O_ROWMAJOR</td> <td>Display the menu in row major order.</td> </tr> <tr> <td>O_IGNORECASE</td> <td>Ignore the case when pattern matching.</td> </tr> <tr> <td>O_SHOWMATCH</td> <td>Place the cursor within the item name when pattern matching.</td> </tr> <tr> <td>O_NONCYCLIC</td> <td>Make certain menu driver requests non-cyclic.</td> </tr> </table> | O_ONEVALUE | Only one item can be selected from the menu. | O_SHOWDESC | Display the description of the items. | O_ROWMAJOR | Display the menu in row major order. | O_IGNORECASE | Ignore the case when pattern matching. | O_SHOWMATCH | Place the cursor within the item name when pattern matching. | O_NONCYCLIC | Make certain menu driver requests non-cyclic. |
| O_ONEVALUE | Only one item can be selected from the menu. | | | | | | | | | | | | |
| O_SHOWDESC | Display the description of the items. | | | | | | | | | | | | |
| O_ROWMAJOR | Display the menu in row major order. | | | | | | | | | | | | |
| O_IGNORECASE | Ignore the case when pattern matching. | | | | | | | | | | | | |
| O_SHOWMATCH | Place the cursor within the item name when pattern matching. | | | | | | | | | | | | |
| O_NONCYCLIC | Make certain menu driver requests non-cyclic. | | | | | | | | | | | | |
| RETURN VALUES | <p>Except for menu_opts (), these routines return one of the following:</p> <table border="0"> <tr> <td>E_OK</td> <td>The routine returned successfully.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> <tr> <td>E_POSTED</td> <td>The menu is already posted.</td> </tr> </table> | E_OK | The routine returned successfully. | E_SYSTEM_ERROR | System error. | E_POSTED | The menu is already posted. | | | | | | |
| E_OK | The routine returned successfully. | | | | | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | | | | | |
| E_POSTED | The menu is already posted. | | | | | | | | | | | | |
| ATTRIBUTES | <p>See attributes(5) for descriptions of the following attributes:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | | | | | | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | | | | | | | |
| MT-Level | Unsafe | | | | | | | | | | | | |
| SEE ALSO | curses(3CURSES), menus(3CURSES), attributes(5) | | | | | | | | | | | | |
| NOTES | The header <menu.h> automatically includes the headers <eti.h> and <curses.h>. | | | | | | | | | | | | |

NAME menu_attributes, set_menu_fore, menu_fore, set_menu_back, menu_back, set_menu_grey, menu_grey, set_menu_pad, menu_pad – control menus display attributes

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

int set_menu_fore(MENU *menu, chtype attr);
chtype menu_fore(MENU *menu);

int set_menu_back(MENU *menu, chtype attr);
chtype menu_back(MENU *menu);

int set_menu_grey(MENU*menu, chtype attr);
chtype menu_grey(MENU *menu);

int set_menu_pad(MENU *menu, int pad);
int menu_pad(MENU *menu);
```

DESCRIPTION

set_menu_fore() sets the foreground attribute of *menu* — the display attribute for the current item (if selectable) on single-valued menus and for selected items on multi-valued menus. This display attribute is a curses library visual attribute. menu_fore() returns the foreground attribute of *menu*.

set_menu_back() sets the background attribute of *menu* — the display attribute for unselected, yet selectable, items. This display attribute is a curses library visual attribute.

set_menu_grey() sets the grey attribute of *menu* — the display attribute for nonselectable items in multi-valued menus. This display attribute is a curses library visual attribute. menu_grey() returns the grey attribute of *menu*.

The pad character is the character that fills the space between the name and description of an item. set_menu_pad() sets the pad character for *menu* to *pad*. menu_pad() returns the pad character of *menu*.

RETURN VALUES These routines return one of the following:

| | |
|----------------|--|
| E_OK | The routine returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

menu_pad(3CURSES)

SEE ALSO | curses(3CURSES), menus(3CURSES), attributes(5)

NOTES | The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

menu_post(3CURSES)

| NAME | menu_post, post_menu, unpost_menu – write or erase menus from associated subwindows | | | | | | | | | | | | | | | | |
|----------------------|--|----------------|------------------------------------|----------------|---------------|----------------|--|----------|-----------------------------|-------------|--|-----------|---|--------------|-------------------------------|-----------------|-------------------------------------|
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int post_menu (MENU *menu) ; int unpost_menu (MENU *menu) ;</pre> | | | | | | | | | | | | | | | | |
| DESCRIPTION | <p>post_menu () writes <i>menu</i> to the subwindow. The application programmer must use curses library routines to display the menu on the physical screen or call update_panels () if the panels library is being used.</p> <p>unpost_menu () erases <i>menu</i> from its associated subwindow.</p> | | | | | | | | | | | | | | | | |
| RETURN VALUES | <p>These routines return one of the following:</p> <table><tr><td>E_OK</td><td>The routine returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr><tr><td>E_BAD_ARGUMENT</td><td>An incorrect argument was passed to the routine.</td></tr><tr><td>E_POSTED</td><td>The menu is already posted.</td></tr><tr><td>E_BAD_STATE</td><td>The routine was called from an initialization or termination function.</td></tr><tr><td>E_NO_ROOM</td><td>The menu does not fit within its subwindow.</td></tr><tr><td>E_NOT_POSTED</td><td>The menu has not been posted.</td></tr><tr><td>E_NOT_CONNECTED</td><td>No items are connected to the menu.</td></tr></table> | E_OK | The routine returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An incorrect argument was passed to the routine. | E_POSTED | The menu is already posted. | E_BAD_STATE | The routine was called from an initialization or termination function. | E_NO_ROOM | The menu does not fit within its subwindow. | E_NOT_POSTED | The menu has not been posted. | E_NOT_CONNECTED | No items are connected to the menu. |
| E_OK | The routine returned successfully. | | | | | | | | | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | | | | | | | | | |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. | | | | | | | | | | | | | | | | |
| E_POSTED | The menu is already posted. | | | | | | | | | | | | | | | | |
| E_BAD_STATE | The routine was called from an initialization or termination function. | | | | | | | | | | | | | | | | |
| E_NO_ROOM | The menu does not fit within its subwindow. | | | | | | | | | | | | | | | | |
| E_NOT_POSTED | The menu has not been posted. | | | | | | | | | | | | | | | | |
| E_NOT_CONNECTED | No items are connected to the menu. | | | | | | | | | | | | | | | | |
| ATTRIBUTES | <p>See attributes(5) for descriptions of the following attributes:</p> <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | | | | | | | | | | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | | | | | | | | | | | |
| MT-Level | Unsafe | | | | | | | | | | | | | | | | |
| SEE ALSO | curses(3CURSES), menus(3CURSES), panels(3CURSES), attributes(5) | | | | | | | | | | | | | | | | |
| NOTES | The header <menu.h> automatically includes the headers <eti.h> and <curses.h>. | | | | | | | | | | | | | | | | |

| NAME | menus – character based menus package | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|-------------------|------------------|---------------------------|------------------------------------|------------------------|--------------------------------|------------------------|---------------------------|-------------------------|-----------------------------|-------------------------------|---------------------------------|-------------------------|------------------------------------|------------------------|----------------------------|------------------------|---------------------------------|------------------------|---------------------------------|----------------------------|---------------------------------|---------------------------|---------------------------------|------------------------|----------------------------|
| SYNOPSIS | <code>#include <menu.h></code> | | | | | | | | | | | | | | | | | | | | | | | | | | |
| DESCRIPTION | <p>The menu library is built using the <code>curses</code> library, and any program using menu routines must call one of the <code>curses</code> initialization routines, such as <code>initscr</code>. A program using these routines must be compiled with <code>-lmenu</code> and <code>-lcurses</code> on the <code>cc</code> command line.</p> <p>The menu package gives the applications programmer a terminal-independent method of creating and customizing menus for user interaction. The menu package includes: item routines, which are used to create and customize menu items; and menu routines, which are used to create and customize menus, assign pre- and post-processing routines, and display and interact with menus.</p> | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Current Default Values for Item Attributes | <p>The menu package establishes initial current default values for item attributes. During item initialization, each item attribute is assigned the current default value for that attribute. An application can change or retrieve a current default attribute value by calling the appropriate set or retrieve routine with a <code>NULL</code> item pointer. If an application changes a current default item attribute value, subsequent items created using <code>new_item()</code> will have the new default attribute value. The attributes of previously created items are not changed if a current default attribute value is changed.</p> | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Routine Name Index | <p>The following table lists each menu routine and the name of the manual page on which it is described.</p> <table border="0" style="width: 100%;"> <thead> <tr> <th style="text-align: left;">Menu Routine Name</th> <th style="text-align: left;">Manual Page Name</th> </tr> </thead> <tbody> <tr> <td><code>current_item</code></td> <td><code>menu_item_current(3X)</code></td> </tr> <tr> <td><code>free_item</code></td> <td><code>menu_item_new(3X)</code></td> </tr> <tr> <td><code>free_menu</code></td> <td><code>menu_new(3X)</code></td> </tr> <tr> <td><code>item_count</code></td> <td><code>menu_items(3X)</code></td> </tr> <tr> <td><code>item_description</code></td> <td><code>menu_item_name(3X)</code></td> </tr> <tr> <td><code>item_index</code></td> <td><code>menu_item_current(3X)</code></td> </tr> <tr> <td><code>item_init</code></td> <td><code>menu_hook(3X)</code></td> </tr> <tr> <td><code>item_name</code></td> <td><code>menu_item_name(3X)</code></td> </tr> <tr> <td><code>item_opts</code></td> <td><code>menu_item_opts(3X)</code></td> </tr> <tr> <td><code>item_opts_off</code></td> <td><code>menu_item_opts(3X)</code></td> </tr> <tr> <td><code>item_opts_on</code></td> <td><code>menu_item_opts(3X)</code></td> </tr> <tr> <td><code>item_term</code></td> <td><code>menu_hook(3X)</code></td> </tr> </tbody> </table> | Menu Routine Name | Manual Page Name | <code>current_item</code> | <code>menu_item_current(3X)</code> | <code>free_item</code> | <code>menu_item_new(3X)</code> | <code>free_menu</code> | <code>menu_new(3X)</code> | <code>item_count</code> | <code>menu_items(3X)</code> | <code>item_description</code> | <code>menu_item_name(3X)</code> | <code>item_index</code> | <code>menu_item_current(3X)</code> | <code>item_init</code> | <code>menu_hook(3X)</code> | <code>item_name</code> | <code>menu_item_name(3X)</code> | <code>item_opts</code> | <code>menu_item_opts(3X)</code> | <code>item_opts_off</code> | <code>menu_item_opts(3X)</code> | <code>item_opts_on</code> | <code>menu_item_opts(3X)</code> | <code>item_term</code> | <code>menu_hook(3X)</code> |
| Menu Routine Name | Manual Page Name | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>current_item</code> | <code>menu_item_current(3X)</code> | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>free_item</code> | <code>menu_item_new(3X)</code> | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>free_menu</code> | <code>menu_new(3X)</code> | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>item_count</code> | <code>menu_items(3X)</code> | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>item_description</code> | <code>menu_item_name(3X)</code> | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>item_index</code> | <code>menu_item_current(3X)</code> | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>item_init</code> | <code>menu_hook(3X)</code> | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>item_name</code> | <code>menu_item_name(3X)</code> | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>item_opts</code> | <code>menu_item_opts(3X)</code> | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>item_opts_off</code> | <code>menu_item_opts(3X)</code> | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>item_opts_on</code> | <code>menu_item_opts(3X)</code> | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <code>item_term</code> | <code>menu_hook(3X)</code> | | | | | | | | | | | | | | | | | | | | | | | | | | |

menus(3CURSES)

| Menus Routine Name | Manual Page Name |
|---------------------------|-------------------------|
| item_userptr | menu_item_userptr (3X) |
| item_value | menu_item_value (3X) |
| item_visible | menu_item_visible (3X) |
| menu_back | menu_attributes (3X) |
| menu_driver | menu_driver (3X) |
| menu_fore | menu_attributes (3X) |
| menu_format | menu_format (3X) |
| menu_grey | menu_attributes (3X) |
| menu_init | menu_hook (3X) |
| menu_items | menu_items (3X) |
| menu_mark | menu_mark (3X) |
| menu_opts | menu_opts (3X) |
| menu_opts_off | menu_opts (3X) |
| menu_opts_on | menu_opts (3X) |
| menu_pad | menu_attributes (3X) |
| menu_pattern | menu_pattern (3X) |
| menu_sub | menu_win (3X) |
| menu_term | menu_hook (3X) |
| menu_userptr | menu_userptr (3X) |
| menu_win | menu_win (3X) |
| new_item | menu_item_new (3X) |
| new_menu | menu_new (3X) |
| pos_menu_cursor | menu_cursor (3X) |
| post_menu | menu_post (3X) |
| scale_menu | menu_win (3X) |
| set_current_item | menu_item_current (3X) |
| set_item_init | menu_hook (3X) |
| set_item_opts | menu_item_opts (3X) |
| set_item_term | menu_hook (3X) |

menus(3CURSES)

| Menus Routine Name | Manual Page Name |
|--------------------|------------------------|
| set_item_userptr | menu_item_userptr (3X) |
| set_item_value | menu_item_value (3X) |
| set_menu_back | menu_attributes (3X) |
| set_menu_fore | menu_attributes (3X) |
| set_menu_format | menu_format (3X) |
| set_menu_grey | menu_attributes (3X) |
| set_menu_init | menu_hook (3X) |
| set_menu_items | menu_items (3X) |
| set_menu_mark | menu_mark (3X) |
| set_menu_opts | menu_opts (3X) |
| set_menu_pad | menu_attributes (3X) |
| set_menu_pattern | menu_pattern (3X) |
| set_menu_sub | menu_win (3X) |
| set_menu_term | menu_hook (3X) |
| set_menu_userptr | menu_userptr (3X) |
| set_menu_win | menu_win (3X) |
| set_top_row | menu_item_current (3X) |
| top_row | menu_item_current (3X) |
| unpost_menu | menu_post (3X) |

RETURN VALUES

Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:

| | |
|----------------|--|
| E_OK | The routine returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. |
| E_POSTED | The menu is already posted. |
| E_CONNECTED | One or more items are already connected to another menu. |
| E_BAD_STATE | The routine was called from an initialization or termination function. |

menus(3CURSES)

| | |
|-------------------|---|
| E_NO_ROOM | The menu does not fit within its subwindow. |
| E_NOT_POSTED | The menu has not been posted. |
| E_UNKNOWN_COMMAND | An unknown request was passed to the menu driver. |
| E_NO_MATCH | The character failed to match. |
| E_NOT_SELECTABLE | The item cannot be selected. |
| E_NOT_CONNECTED | No items are connected to the menu. |
| E_REQUEST_DENIED | The menu driver could not process the request. |

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3CURSES)`, `attributes(5)`

NOTES The header `<menu.h>` automatically includes the headers `<eti.h>` and `<curses.h>`.

NAME menu_win, set_menu_win, set_menu_sub, menu_sub, scale_menu – menus window and subwindow association routines

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

int set_menu_win(MENU *menu, WINDOW *win);
WINDOW *menu_win(MENU *menu);

int set_menu_sub(MENU *menu, WINDOW *sub);
WINDOW *menu_sub(MENU *menu);

int scale_window(MENU *menu, int *rows, int *cols);
```

DESCRIPTION set_menu_win() sets the window of *menu* to *win*. menu_win() returns a pointer to the window of *menu*. set_menu_sub() sets the subwindow of *menu* to *sub*. menu_sub() returns a pointer to the subwindow of *menu*. scale_window() returns the minimum window size necessary for the subwindow of *menu*. *rows* and *cols* are pointers to the locations used to return the values.

RETURN VALUES Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:

E_OK The routine returned successfully.

E_SYSTEM_ERROR System error.

E_BAD_ARGUMENT An incorrect argument was passed to the routine.

E_POSTED The menu is already posted.

E_NOT_CONNECTED No items are connected to the menu.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), attributes(5)

NOTES The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

menu_term(3CURSES)

| | | | | | |
|----------------------|--|------|------------------------------------|----------------|---------------|
| NAME | menu_hook, set_item_init, item_init, set_item_term, item_term, set_menu_init, menu_init, set_menu_term, menu_term – assign application-specific routines for automatic invocation by menus | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int set_item_init(MENU *menu, void (*func)(MENU *)); int set_item_term(MENU *menu, void (*func)(MENU *)); void item_term(MENU *menu); int set_menu_init(MENU *menu, void (*func)(MENU *)); void menu_init(MENU *menu); int set_menu_term(MENU *menu, void (*func)(MENU *)); void menu_term(MENU *menu);</pre> | | | | |
| DESCRIPTION | <p>set_item_init() assigns the application-defined function to be called when the <i>menu</i> is posted and just after the current item changes. item_init() returns a pointer to the item initialization routine, if any, called when the <i>menu</i> is posted and just after the current item changes.</p> <p>set_item_term() assigns an application-defined function to be called when the <i>menu</i> is unposted and just before the current item changes. item_term() returns a pointer to the termination function, if any, called when the <i>menu</i> is unposted and just before the current item changes.</p> <p>set_menu_init() assigns an application-defined function to be called when the <i>menu</i> is posted and just after the top row changes on a posted menu. menu_init() returns a pointer to the menu initialization routine, if any, called when the <i>menu</i> is posted and just after the top row changes on a posted menu.</p> <p>set_menu_term() assigns an application-defined function to be called when the <i>menu</i> is unposted and just before the top row changes on a posted menu. menu_term() returns a pointer to the menu termination routine, if any, called when the <i>menu</i> is unposted and just before the top row changes on a posted menu.</p> | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <table><tr><td>E_OK</td><td>The routine returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr></table> | E_OK | The routine returned successfully. | E_SYSTEM_ERROR | System error. |
| E_OK | The routine returned successfully. | | | | |
| E_SYSTEM_ERROR | System error. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |

menu_term(3CURSES)

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), attributes(5)

NOTES The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

menu_userptr(3CURSES)

NAME menu_userptr, set_menu_userptr – associate application data with menus

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

char *menu_userptr(MENU *menu);

int set_menu_userptr(MENU *menu, char *userptr);
```

DESCRIPTION Every menu has an associated user pointer that can be used to store relevant information. `set_menu_userptr()` sets the user pointer of `menu`. `menu_userptr()` returns the user pointer of `menu`.

RETURN VALUES `menu_userptr()` returns NULL on error.

`set_menu_userptr()` returns one of the following:

| | |
|----------------|------------------------------------|
| E_OK | The routine returned successfully. |
| E_SYSTEM_ERROR | System error. |

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3CURSES)`, `menus(3CURSES)`, `attributes(5)`

NOTES The header `<menu.h>` automatically includes the headers `<eti.h>` and `<curses.h>`.

NAME menu_win, set_menu_win, set_menu_sub, menu_sub, scale_menu – menus window and subwindow association routines

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

int set_menu_win(MENU *menu, WINDOW *win);
WINDOW *menu_win(MENU *menu);

int set_menu_sub(MENU *menu, WINDOW *sub);
WINDOW *menu_sub(MENU *menu);

int scale_window(MENU *menu, int *rows, int *cols);
```

DESCRIPTION set_menu_win() sets the window of *menu* to *win*. menu_win() returns a pointer to the window of *menu*. set_menu_sub() sets the subwindow of *menu* to *sub*. menu_sub() returns a pointer to the subwindow of *menu*. scale_window() returns the minimum window size necessary for the subwindow of *menu*. *rows* and *cols* are pointers to the locations used to return the values.

RETURN VALUES Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:

E_OK The routine returned successfully.

E_SYSTEM_ERROR System error.

E_BAD_ARGUMENT An incorrect argument was passed to the routine.

E_POSTED The menu is already posted.

E_NOT_CONNECTED No items are connected to the menu.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), attributes(5)

NOTES The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

meta(3CURSES)

| | |
|--------------------|---|
| NAME | <code>curs_inopts</code> , <code>cbreak</code> , <code>nocbreak</code> , <code>echo</code> , <code>noecho</code> , <code>halfdelay</code> , <code>intrflush</code> , <code>keypad</code> , <code>meta</code> , <code>nodelay</code> , <code>notimeout</code> , <code>raw</code> , <code>noraw</code> , <code>noqiflush</code> , <code>qiflush</code> , <code>timeout</code> , <code>wtimeout</code> , <code>typeahead</code> – curses terminal input option control routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int cbreak(void); int nocbreak(void); int echo(void); int noecho(void); int halfdelay(int <i>tenths</i>); int intrflush(WINDOW *<i>win</i>, bool <i>bf</i>); int keypad(WINDOW *<i>win</i>, bool <i>bf</i>); int meta(WINDOW *<i>win</i>, bool <i>bf</i>); int nodelay(WINDOW *<i>win</i>, bool <i>bf</i>); int notimeout(WINDOW *<i>win</i>, bool <i>bf</i>); int raw(void); int noraw(void); void noqiflush(void); void qiflush(void); void timeout(int <i>delay</i>); void wtimeout(WINDOW *<i>win</i>, int <i>delay</i>); int typeahead(int <i>fildes</i>);</pre> |
| DESCRIPTION | <p>The <code>cbreak()</code> and <code>nocbreak()</code> routines put the terminal into and out of <code>cbreak()</code> mode, respectively. In this mode, characters typed by the user are immediately available to the program, and erase/kill character-processing is not performed. When out of this mode, the tty driver buffers the typed characters until a newline or carriage return is typed. Interrupt and flow control characters are unaffected by this mode. Initially the terminal may or may not be in <code>cbreak()</code> mode, as the mode is inherited; therefore, a program should call <code>cbreak()</code> or <code>nocbreak()</code> explicitly. Most interactive programs using curses set the <code>cbreak()</code> mode.</p> <p>Note that <code>cbreak()</code> overrides <code>raw()</code>. (See <code>curs_getch(3CURSES)</code> for a discussion of how these routines interact with <code>echo()</code> and <code>noecho()</code>.)</p> <p>The <code>echo()</code> and <code>noecho()</code> routines control whether characters typed by the user are echoed by <code>getch()</code> as they are typed. Echoing by the tty driver is always disabled, but initially <code>getch()</code> is in echo mode, so characters typed are echoed. Authors of</p> |

most interactive programs prefer to do their own echoing in a controlled area of the screen, or not to echo at all, so they disable echoing by calling `noecho()`. (See `curs_getch(3CURSES)` for a discussion of how these routines interact with `cbreak()` and `nocbreak()`.)

The `halfdelay()` routine is used for half-delay mode, which is similar to `cbreak()` mode in that characters typed by the user are immediately available to the program. However, after blocking for *tenths* tenths of seconds, `ERR` is returned if nothing has been typed. The value of *tenths* must be a number between 1 and 255. Use `nocbreak()` to leave half-delay mode.

If the `intrflush()` option is enabled, (*bf* is `TRUE`), when an interrupt key is pressed on the keyboard (interrupt, break, quit) all output in the tty driver queue will be flushed, giving the effect of faster response to the interrupt, but causing `curses` to have the wrong idea of what is on the screen. Disabling (*bf* is `FALSE`), the option prevents the flush. The default for the option is inherited from the tty driver settings. The window argument is ignored.

The `keypad()` option enables the keypad of the user's terminal. If enabled (*bf* is `TRUE`), the user can press a function key (such as an arrow key) and `wgetch()` returns a single value representing the function key, as in `KEY_LEFT`. If disabled (*bf* is `FALSE`), `curses` does not treat function keys specially and the program has to interpret the escape sequences itself. If the keypad in the terminal can be turned on (made to transmit) and off (made to work locally), turning on this option causes the terminal keypad to be turned on when `wgetch()` is called. The default value for keypad is false.

Initially, whether the terminal returns 7 or 8 significant bits on input depends on the control mode of the tty driver (see `termio(7I)`). To force 8 bits to be returned, invoke `meta(win, TRUE)`. To force 7 bits to be returned, invoke `meta(win, FALSE)`. The window argument, *win*, is always ignored. If the terminfo capabilities `smm` (`meta_on`) and `rmm` (`meta_off`) are defined for the terminal, `smm` is sent to the terminal when `meta(win, TRUE)` is called and `rmm` is sent when `meta(win, FALSE)` is called.

The `nodelay()` option causes `getch()` to be a non-blocking call. If no input is ready, `getch()` returns `ERR`. If disabled (*bf* is `FALSE`), `getch()` waits until a key is pressed.

While interpreting an input escape sequence, `wgetch()` sets a timer while waiting for the next character. If `notimeout(win, TRUE)` is called, then `wgetch()` does not set a timer. The purpose of the timeout is to differentiate between sequences received from a function key and those typed by a user.

With the `raw()` and `noraw()` routines, the terminal is placed into or out of raw mode. Raw mode is similar to `cbreak()` mode, in that characters typed are immediately passed through to the user program. The differences are that in raw mode, the interrupt, quit, suspend, and flow control characters are all passed through uninterpreted, instead of generating a signal. The behavior of the `BREAK` key depends on other bits in the tty driver that are not set by `curses`.

meta(3CURSES)

When the `noqiflush()` routine is used, normal flush of input and output queues associated with the `INTR`, `QUIT` and `SUSP` characters will not be done (see `termio(7I)`). When `qiflush()` is called, the queues will be flushed when these control characters are read.

The `timeout()` and `wtimeout()` routines set blocking or non-blocking read for a given window. If *delay* is negative, blocking read is used (that is, waits indefinitely for input). If *delay* is zero, then non-blocking read is used (that is, read returns `ERR` if no input is waiting). If *delay* is positive, then read blocks for *delay* milliseconds, and returns `ERR` if there is still no input. Hence, these routines provide the same functionality as `nodelay()`, plus the additional capability of being able to block for only *delay* milliseconds (where *delay* is positive).

`curses` does "line-breakout optimization" by looking for typeahead periodically while updating the screen. If input is found, and it is coming from a `tty`, the current update is postponed until `refresh()` or `doupdate()` is called again. This allows faster response to commands typed in advance. Normally, the input `FILE` pointer passed to `newterm()`, or `stdin` in the case that `initscr()` was used, will be used to do this typeahead checking. The `typeahead()` routine specifies that the file descriptor *fildev* is to be used to check for typeahead instead. If *fildev* is `-1`, then no typeahead checking is done.

RETURN VALUES

All routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_getch(3CURSES)`, `curs_initscr(3CURSES)`, `curses(3CURSES)`, `attributes(5)`, `termio(7I)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `echo()`, `noecho()`, `halfdelay()`, `intrflush()`, `meta()`, `nodelay()`, `notimeout()`, `noqiflush()`, `qiflush()`, `timeout()`, and `wtimeout()` may be macros.

| NAME | meta – enable/disable meta keys | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int meta(WINDOW *win, bool bf);</pre> | | | | | | |
| PARAMETERS | <p><i>win</i> Is an ignored parameter.</p> <p><i>bf</i> Is a Boolean expression.</p> | | | | | | |
| DESCRIPTION | <p>Whether a terminal returns 7 or 8 significant bits initially depends on the control mode of the terminal driver. The <code>meta()</code> function forces the number of bits to be returned by <code>getch(3XCURSES)</code> to be 7 (if <i>bf</i> is FALSE) or 8 (if <i>bf</i> is TRUE).</p> <p>If the program handling the data can only pass 7-bit characters or strips the 8th bit, 8 bits cannot be handled.</p> <p>If the terminfo capabilities <code>smm</code> (<code>meta_on</code>) and <code>rmm</code> (<code>meta_off</code>) are defined for the terminal, <code>smm</code> is sent to the terminal when <code>meta(win, TRUE)</code> is called, and <code>rmm</code> is sent when <code>meta(win, FALSE)</code> is called.</p> <p>This function is useful when extending the non-text command set in applications where the META key is used.</p> | | | | | | |
| RETURN VALUES | On success, the <code>meta()</code> function returns OK. Otherwise, it returns ERR. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>getch(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

move(3CURSES)

NAME curs_ move, move, wmove – move curses window cursor

SYNOPSIS

```
cc [ flag ... ] file ... -lcurses [ library ... ]
#include <curses.h>

int move(int y, int x);
int wmove(WINDOW *win, int y, int x);
```

DESCRIPTION With these routines, the cursor associated with the window is moved to line *y* and column *x*. This routine does not move the physical cursor of the terminal until `refresh()` is called. The position specified is relative to the upper left-hand corner of the window, which is (0,0).

RETURN VALUES These routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_refresh(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `move()` may be a macro.

| | |
|--------------------|--|
| NAME | plot, arc, box, circle, closepl, closevt, cont, erase, label, line, linemod, move, openpl, openvt, point, space – graphics interface |
| SYNOPSIS | <pre>cc [flag ...] file ... -lplot [library...] #include <plot.h> void arc(short x0, short y0, short x1, short y1, short x2, short y2); void box(short x0, short y0, short x1, short y1); void circle(short x, short y, short r); void closepl(); void closevt(); void cont(short x, short y); void erase(); void label(char *s); void line(short x0, short y0, short x1, short y1); void linemod(char *s); void move(short x, short y); void openpl(); void openvt(); void point(short x, short y); void space(short x0, short y0, short x1, short y1);</pre> |
| DESCRIPTION | <p>These functions generate graphics output for a set of output devices. The format of the output is dependent upon which link editor option is used when the program is compiled and linked (see Link Editor).</p> <p>The term "current point" refers to the current setting for the x and y coordinates.</p> <p>The <code>arc()</code> function specifies a circular arc. The coordinates $(x0, y0)$ specify the center of the arc. The coordinates $(x1, y1)$ specify the starting point of the arc. The coordinates $(x2, y2)$ specify the end point of the circular arc.</p> <p>The <code>box()</code> function specifies a rectangle with coordinates $(x0, y0)$, $(x0, y1)$, $(x1, y0)$, and $(x1, y1)$. The current point is set to $(x1, y1)$.</p> <p>The <code>circle()</code> function specifies a circle with a center at the coordinates (x, y) and a radius of r.</p> <p>The <code>closevt()</code> and <code>closepl()</code> functions flush the output.</p> <p>The <code>cont()</code> function specifies a line beginning at the current point and ending at the coordinates (x, y). The current point is set to (x, y).</p> |

move(3PLOT)

The `erase()` function starts another frame of output.

The `label()` function places the null terminated string `s` so that the first character falls on the current point. The string is then terminated by a NEWLINE character.

The `line()` function draws a line starting at the coordinates (x_0, y_0) and ending at the coordinates (x_1, y_1) . The current point is set to (x_1, y_1) .

The `linemod()` function specifies the style for drawing future lines. `s` may contain one of the following: `dotted`, `solid`, `longdashed`, `shortdashed`, or `dotdashed`.

The `move()` function sets the current point to the coordinates (x, y) .

The `openpl()` or `openvt()` function must be called to open the device before any other plot functions are called.

The `point()` function plots the point given by the coordinates (x, y) . The current point is set to (x, y) .

The `space()` function specifies the size of the plotting area. The plot will be reduced or enlarged as necessary to fit the area specified. The coordinates (x_0, y_0) specify the lower left hand corner of the plotting area. The coordinates (x_1, y_1) specify the upper right hand corner of the plotting area.

Link Editor Various flavors of these functions exist for different output devices. They are obtained by using the following `ld(1)` options:

| | |
|---------------------|--|
| <code>-lplot</code> | device-independent graphics stream on standard output in the format described in <code>plot(4B)</code> |
| <code>-l300</code> | GSI 300 terminal |
| <code>-l300s</code> | GSI 300S terminal |
| <code>-l4014</code> | Tektronix 4014 terminal |
| <code>-l450</code> | GSI 450 terminal |
| <code>-lvt0</code> | |

FILES

| | |
|--|----------------------|
| <code>/usr/lib/libplot.a</code> | archive library |
| <code>/usr/lib/libplot.so.1</code> | shared object |
| <code>/usr/lib/sparcv9/libplot.so.1</code> | 64-bit shared object |
| <code>/usr/lib/lib300.a</code> | archive library |
| <code>/usr/lib/lib300.so.1</code> | shared object |

```

/usr/lib/sparcv9/lib300.so.1
  64-bit shared object
/usr/lib/lib300s.a
  archive library
/usr/lib/lib300s.so.1
  shared object
/usr/lib/sparcv9/lib300s.so.1
  64-bit shared object
/usr/lib/lib4014.a
  archive library
/usr/lib/lib4014.so.1
  shared object
/usr/lib/sparcv9/lib4014.so.1
  64-bit shared object
/usr/lib/lib450.a
  archive library
/usr/lib/lib450.so.1
  shared object
/usr/lib/sparcv9/lib450.so.1
  64-bit shared object
/usr/lib/libvt0.a
  archive library
/usr/lib/libvt0.so.1
  shared object
/usr/lib/sparcv9/libvt0.so.1
  64-bit shared object

```

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `graph(1)`, `ld(1)`, `libplot(3LIB)`, `plot(4B)`, `attributes(5)`

move(3XCURSES)

NAME move, wmove – move cursor in window

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

int move(int y, int x);

int wmove(WINDOW *win, int y, int x);
```

PARAMETERS

| | |
|------------|---|
| <i>y</i> | Is the y (row) coordinate of the position of the cursor in the window. |
| <i>x</i> | Is the x (column) coordinate of the position of the cursor in the window. |
| <i>win</i> | Is a pointer to the window in which the cursor is to be written. |

DESCRIPTION The `move()` function moves the logical cursor (for `stdscr`) to the position specified by *y* (row) and *x* (column), where the upper left corner of the window is row 0, column 0. The `wmove()` function performs the same action, but moves the cursor in the window specified by *win*. The physical cursor will not move until after a call to `refresh(3XCURSES)` or `doupdate(3XCURSES)`.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `doupdate(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

NAME form_field, set_form_fields, form_fields, field_count, move_field – connect fields to forms

SYNOPSIS

```
cc [ flag... ] file... -lform -lcurses [ library... ]
#include <form.h>

int set_form_fields(FORM *form, FIELD **field);
FIELD **form_fields(FORM *form);
int field_count(FORM *form);
int move_field(FIELD *field, int frow, int fcol);
```

DESCRIPTION

set_form_fields() changes the fields connected to *form* to *fields*. The original fields are disconnected.

form_fields() returns a pointer to the field pointer array connected to *form*.

field_count() returns the number of fields connected to *form*.

move_field() moves the disconnected *field* to the location *frow*, *fcol* in the forms subwindow.

RETURN VALUES

form_fields() returns NULL on error.

field_count() returns -1 on error.

set_form_fields() and move_field() return one of the following:

| | |
|----------------|---|
| E_OK | The function returned successfully. |
| E_CONNECTED | The field is already connected to a form. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An argument is incorrect |
| E_POSTED | The form is posted. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), forms(3CURSES), attributes(5)

NOTES The header <form.h> automatically includes the headers <eti.h> and <curses.h>.

movenextch(3CURSES)

NAME | curs_alecompat, movenextch, wmovenextch, moveprevch, wmoveprevch, adjcurspos, wadjcurspos – these functions are added to ALE curses library for moving the cursor by character.

SYNOPSIS |

```
cc [ flag ... ] file ... -lcurses [ library .. ]
#include <curses.h>

int movenextch(void);
int wmovenextch(WINDOW *win);
int moveprevch(void);
int wmoveprevch(WINDOW *win);
int adjcurspos(void);
int wadjcurspos(WINDOW *win);
```

DESCRIPTION |

movenextch() and wmovenextch() move the cursor to the next character to the right. If the next character is a multicolumn character, the cursor is positioned on the first (left-most) column of that character. The new cursor position will be on the next character, even if the cursor was originally positioned on the left-most column of a multicolumn character. Note that the simple cursor increment (++x) does not guarantee movement to the next character, if the cursor was originally positioned on a multicolumn character. getyx(3CURSES) can be used to find the new position.

moveprevc() and wmoveprevch() routines are the opposite of movenextch() and wmovenextch(), moving the cursor to the left-most column of the previous character.

adjcurspos() and wadjcurspos() move the cursor to the first(left-most) column of the multicolumn character that the cursor is presently on. If the cursor is already on the first column, or if the cursor is on a single-column character, these routines will have no effect.

RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | curses(3CURSES), getyx(3CURSES), attributes(5)

NOTES | The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <widec.h>.

Note that movenextch(), moveprevch(), and adjcurspos() may be macros.

move_panel(3CURSES)

NAME panel_move, move_panel – move a panels window on the virtual screen

SYNOPSIS

```
cc [ flag ... ] file ... -lpanel -lcurses [ library .. ]
#include <panel.h>

int move_panel (PANEL *panel, int starty, int startx);
```

DESCRIPTION move_panel () moves the curses window associated with *panel* so that its upper left-hand corner is at *starty*, *startx*. See usage note, below.

RETURN VALUES OK is returned if the routine completes successfully, otherwise ERR is returned.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), panel_update(3CURSES), panels(3CURSES), attributes(5)

NOTES For panels windows, use move_panel () instead of the mvwin () curses routine. Otherwise, update_panels () will not properly update the virtual screen.

The header <panel.h> automatically includes the header <curses.h>.

moveprevch(3CURSES)

| NAME | <code>curs_alecompat</code> , <code>movenextch</code> , <code>wmovenextch</code> , <code>moveprevch</code> , <code>wmoveprevch</code> , <code>adjcurspos</code> , <code>wadjcurspos</code> – these functions are added to ALE curses library for moving the cursor by character. | | | | |
|---------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int movenextch(void); int wmovenextch(WINDOW *win); int moveprevch(void); int wmoveprevch(WINDOW *win); int adjcurspos(void); int wadjcurspos(WINDOW *win);</pre> | | | | |
| DESCRIPTION | <p><code>movenextch()</code> and <code>wmovenextch()</code> move the cursor to the next character to the right. If the next character is a multicolumn character, the cursor is positioned on the first (left-most) column of that character. The new cursor position will be on the next character, even if the cursor was originally positioned on the left-most column of a multicolumn character. Note that the simple cursor increment (<code>++x</code>) does not guarantee movement to the next character, if the cursor was originally positioned on a multicolumn character. <code>getyx(3CURSES)</code> can be used to find the new position.</p> <p><code>moveprevch()</code> and <code>wmoveprevch()</code> routines are the opposite of <code>movenextch()</code> and <code>wmovenextch()</code>, moving the cursor to the left-most column of the previous character.</p> <p><code>adjcurspos()</code> and <code>wadjcurspos()</code> move the cursor to the first(left-most) column of the multicolumn character that the cursor is presently on. If the cursor is already on the first column, or if the cursor is on a single-column character, these routines will have no effect.</p> | | | | |
| RETURN VALUE | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>getyx(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header file <code><curses.h></code> automatically includes the header files <code><stdio.h></code> , <code><unctrl.h></code> and <code><widec.h></code> . | | | | |
| | Note that <code>movenextch()</code> , <code>moveprevch()</code> , and <code>adjcurspos()</code> may be macros. | | | | |

| | |
|----------------------|--|
| NAME | curs_addch, addch, waddch, mvaddch, mvwaddch, echochar, wechochar – add a character (with attributes) to a curses window and advance cursor |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int addch(chtype ch); int waddch(WINDOW *win, chtype ch); int mvaddch(int y, int x, chtype ch); int mvwaddch(WINDOW *win, int y, int x, chtype ch); int echochar(chtype ch); int wechochar(WINDOW *win, chtype ch);</pre> |
| DESCRIPTION | <p>With the <code>addch()</code>, <code>waddch()</code>, <code>mvaddch()</code>, and <code>mvwaddch()</code> routines, the character <code>ch</code> is put into the window at the current cursor position of the window and the position of the window cursor is advanced. Its function is similar to that of <code>putchar()</code>. At the right margin, an automatic newline is performed. At the bottom of the scrolling region, if <code>scrollok()</code> is enabled, the scrolling region is scrolled up one line.</p> <p>If <code>ch</code> is a tab, newline, or backspace, the cursor is moved appropriately within the window. A newline also does a <code>clrtoeol()</code> before moving. Tabs are considered to be at every eighth column. If <code>ch</code> is another control character, it is drawn in the <code>^X</code> notation. Calling <code>winch()</code> after adding a control character does not return the control character, but instead returns the representation of the control character. See <code>curs_inch(3CURSES)</code>.</p> <p>Video attributes can be combined with a character by OR-ing them into the parameter. This results in these attributes also being set. (The intent here is that text, including attributes, can be copied from one place to another using <code>inch()</code> and <code>addch()</code>.) (see <code>standout()</code>, predefined video attribute constants, on the <code>curs_attr(3CURSES)</code> page).</p> <p>The <code>echochar()</code> and <code>wechochar()</code> routines are functionally equivalent to a call to <code>addch()</code> followed by a call to <code>refresh()</code>, or a call to <code>waddch</code> followed by a call to <code>wrefresh()</code>. The knowledge that only a single character is being output is taken into consideration and, for non-control characters, a considerable performance gain might be seen by using these routines instead of their equivalents.</p> |
| Line Graphics | <p>The following variables may be used to add line drawing characters to the screen with routines of the <code>addch()</code> family. When variables are defined for the terminal, the <code>A_ALTCHARSET</code> bit is turned on (see <code>curs_attr(3CURSES)</code>). Otherwise, the default character listed below is stored in the variable. The names chosen are consistent with the VT100 nomenclature.</p> |

mvaddch(3CURSES)

| Name | Default | Glyph Description |
|--------------|---------|-------------------------|
| ACS_ULCORNER | + | upper left-hand corner |
| ACS_LLCORNER | + | lower left-hand corner |
| ACS_URCORNER | + | upper right-hand corner |
| ACS_LRCORNER | + | lower right-hand corner |
| ACS_RTEE | + | right tee |
| ACS_LTEE | + | left tee |
| ACS_BTEE | + | bottom tee |
| ACS_TTEE | + | top tee |
| ACS_HLINE | - | horizontal line |
| ACS_VLINE | | vertical line |
| ACS_PLUS | + | plus |
| ACS_S1 | - | scan line 1 |
| ACS_S9 | - | scan line 9 |
| ACS_DIAMOND | + | diamond |
| ACS_CKBOARD | : | checker board (stipple) |
| ACS_DEGREE | ' | degree symbol |
| ACS_PLMINUS | # | plus/minus |
| ACS_BULLET | o | bullet |
| ACS_LARROW | < | arrow pointing left |
| ACS_RARROW | > | arrow pointing right |
| ACS_DARROW | v | arrow pointing down |
| ACS_UARROW | ^ | arrow pointing up |
| ACS_BOARD | # | board of squares |
| ACS_LANTERN | # | lantern symbol |
| ACS_BLOCK | # | solid square block |

RETURN VALUES

All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

mvaddch(3CURSES)

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_attr(3CURSES), curs_clear(3CURSES), curs_inch(3CURSES), curs_outopts(3CURSES), curs_refresh(3CURSES), curses(3CURSES), putc(3C), attributes(5)

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `addch()`, `mvaddch()`, `mvwaddch()`, and `echochar()` may be macros.

mvaddch(3XCURSES)

| | | | | | | | | | | | |
|----------------------|---|---------------|---|----------|--|----------|--|----------|---|------------|---|
| NAME | addch, mvaddch, mvwaddch, waddch – add a character (with rendition) to a window | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addch(const chtype ch); int mvaddch(int y, int x, const chtype ch); int mvwaddch(WINDOW *win, int y, int x, const chtype ch); int waddch(WINDOW *win, const chtype ch);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The addch() function writes a character to the stdscr window at the current cursor position. The mvaddch() and mvwaddch() functions write the character to the position indicated by the x (column) and y (row) parameters. The mvaddch() function writes the character to the stdscr window, while mvwaddch() writes the character to the window specified by win. The waddch() function is identical to addch(), but writes the character to the window specified by win.</p> <p>These functions advance the cursor after writing the character. Characters that do not fit on the end of the current line are wrapped to the beginning of the next line unless the current line is the last line of the window and scrolling is disabled. In that situation, characters which extend beyond the end of the line are discarded.</p> <p>When ch is a backspace, carriage return, newline, or tab, X/Open Curses moves the cursor appropriately. Each tab character moves the cursor to the next tab stop. By default, tab stops occur every eight columns. When ch is a control character other than backspace, carriage return, newline, or tab, it is written using ^x notation, where x is a printable character. When X/Open Curses writes ch to the last character position on a line, it automatically generates a newline. When ch is written to the last character position of a scrolling region and scrolllok() is enabled, X/Open Curses scrolls the scrolling region up one line (see clearok(3XCURSES)).</p> | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>wchstr</i></td><td>Is a pointer to the <code>cchar_t</code> string to be copied to the window.</td></tr><tr><td><i>n</i></td><td>Is the maximum number of characters to be copied from <i>wchstr</i>. If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line.</td></tr><tr><td><i>y</i></td><td>Is the y (row) coordinate of the starting position of <i>wchstr</i> in the window.</td></tr><tr><td><i>x</i></td><td>Is the x (column) coordinate of the starting position of <i>wchstr</i> in the window.</td></tr><tr><td><i>win</i></td><td>Is a pointer to the window to which the string is to be copied.</td></tr></table> | <i>wchstr</i> | Is a pointer to the <code>cchar_t</code> string to be copied to the window. | <i>n</i> | Is the maximum number of characters to be copied from <i>wchstr</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | <i>y</i> | Is the y (row) coordinate of the starting position of <i>wchstr</i> in the window. | <i>x</i> | Is the x (column) coordinate of the starting position of <i>wchstr</i> in the window. | <i>win</i> | Is a pointer to the window to which the string is to be copied. |
| <i>wchstr</i> | Is a pointer to the <code>cchar_t</code> string to be copied to the window. | | | | | | | | | | |
| <i>n</i> | Is the maximum number of characters to be copied from <i>wchstr</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | | | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of the starting position of <i>wchstr</i> in the window. | | | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of the starting position of <i>wchstr</i> in the window. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window to which the string is to be copied. | | | | | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | | | | | |

mvaddch(3XCURSES)

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attroff(3XCURSES)`, `bkgdset(3XCURSES)`, `douupdate(3XCURSES)`, `inch(3XCURSES)`, `insch(3XCURSES)`, `libcurses(3XCURSES)`, `nl(3XCURSES)`, `printw(3XCURSES)`, `scrollok(3XCURSES)`, `scl(3XCURSES)`, `terminfo(4)`, `attributes(5)`, `standards(5)`

mvaddchnstr(3CURSES)

| NAME | curs_addchstr, addchstr, addchnstr, waddchstr, waddchnstr, mvaddchstr, mvaddchnstr, mvwaddchstr, mvwaddchnstr – add string of characters and attributes to a curses window | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int addchstr(chtype *chstr); int addchnstr(chtype *chstr, int n); int waddchstr(WINDOW *win, chtype *chstr); int waddchnstr(WINDOW *win, chtype *chstr, int n); int mvaddchstr(int y, int x, chtype *chstr); int mvaddchnstr(int y, int x, chtype *chstr, int n); int mvwaddchstr(WINDOW *win, int y, int x, chtype *chstr); int mvwaddchnstr(WINDOW *win, int y, int x, chtype *chstr, int n);</pre> | | | | |
| DESCRIPTION | <p>All of these routines copy <i>chstr</i> directly into the window image structure starting at the current cursor position. The four routines with <i>n</i> as the last argument copy at most <i>n</i> elements, but no more than will fit on the line. If <i>n</i> = -1 then the whole string is copied, to the maximum number that fit on the line.</p> <p>The position of the window cursor is not advanced. These routines works faster than <code>waddnstr()</code> (see <code>curs_addstr(3CURSES)</code>) because they merely copy <i>chstr</i> into the window image structure. On the other hand, care must be taken when using these functions because they do not perform any kind of checking (such as for the newline character), they do not advance the current cursor position, and they truncate the string, rather than wrapping it around to the next line.</p> | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_addstr(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that all routines except <code>waddchnstr()</code> and <code>waddchstr()</code> may be macros.</p> | | | | |

| | |
|----------------------|---|
| NAME | addchstr, addchnstr, mvaddchstr, mvaddchnstr, mvwaddchstr, mvwaddchnstr, waddchstr, waddchnstr – copy a character string (with renditions) to a window |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addchstr(const chtype *chstr); int addchnstr(const chtype *chstr, int n); int mvaddchnstr(int y, int x, const chtype *chstr, int n); int mvaddchstr(int y, int x, const chtype *chstr); int mvwaddchnstr(WINDOW *win, int y, int x, const chtype *chstr, int n); int mvwaddchstr(WINDOW *win, int y, int x, const chtype *chstr); int waddchstr(WINDOW *win, const chtype *chstr); int waddchnstr(WINDOW *win, const chtype *chstr, int n);</pre> |
| DESCRIPTION | <p>The <code>addchstr()</code> function copies the <code>chtype</code> character string to the <code>stdscr</code> window at the current cursor position. The <code>mvaddchstr()</code> and <code>mvwaddchstr()</code> functions copy the character string to the starting position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <code>win</code>). The <code>waddchstr()</code> is identical to <code>addchstr()</code>, but writes to the window specified by <code>win</code>.</p> <p>The <code>addchnstr()</code>, <code>waddchnstr()</code>, <code>mvaddchnstr()</code>, and <code>mvwaddchnstr()</code> functions write <code>n</code> characters to the window, or as many as will fit on the line. If <code>n</code> is less than 0, the entire string is written, or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <code>x</code> and <code>y</code> parameters.</p> <p>These functions differ from the <code>addstr(3XCURSES)</code> set of functions in two important respects. First, these functions do <i>not</i> advance the cursor after writing the string to the window. Second, the current window rendition is not combined with the character; only the attributes that are already part of the <code>chtype</code> character are used.</p> |
| PARAMETERS | <p><code>chstr</code> Is a pointer to the <code>chtype</code> string to be copied to the window.</p> <p><code>n</code> Is the maximum number of characters to be copied from <code>chstr</code>. If <code>n</code> is less than 0, the entire string is written or as much of it as fits on the line.</p> <p><code>y</code> Is the <code>y</code> (row) coordinate of the starting position of <code>chstr</code> in the window.</p> <p><code>x</code> Is the <code>x</code> (column) coordinate of the starting position of <code>chstr</code> in the window.</p> <p><code>win</code> Is a pointer to the window to which the string is to be copied.</p> |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. |

mvaddchnstr(3XCURSES)

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addch(3XCURSES)`, `addnstr(3XCURSES)`, `attroff(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_addchstr, addchstr, addchnstr, waddchstr, waddchnstr, mvaddchstr, mvaddchnstr, mvwaddchstr, mvwaddchnstr – add string of characters and attributes to a curses window | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int addchstr(chtype *chstr); int addchnstr(chtype *chstr, int n); int waddchstr(WINDOW *win, chtype *chstr); int waddchnstr(WINDOW *win, chtype *chstr, int n); int mvaddchstr(int y, int x, chtype *chstr); int mvaddchnstr(int y, int x, chtype *chstr, int n); int mvwaddchstr(WINDOW *win, int y, int x, chtype *chstr); int mvwaddchnstr(WINDOW *win, int y, int x, chtype *chstr, int n);</pre> | | | | |
| DESCRIPTION | <p>All of these routines copy <i>chstr</i> directly into the window image structure starting at the current cursor position. The four routines with <i>n</i> as the last argument copy at most <i>n</i> elements, but no more than will fit on the line. If <i>n</i>=-1 then the whole string is copied, to the maximum number that fit on the line.</p> <p>The position of the window cursor is not advanced. These routines works faster than <code>waddnstr()</code> (see <code>curs_addstr(3CURSES)</code>) because they merely copy <i>chstr</i> into the window image structure. On the other hand, care must be taken when using these functions because they do not perform any kind of checking (such as for the newline character), they do not advance the current cursor position, and they truncate the string, rather than wrapping it around to the next line.</p> | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_addstr(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that all routines except <code>waddchnstr()</code> and <code>waddchstr()</code> may be macros.</p> | | | | |

mvaddchstr(3XCURSES)

| | | | | | | | | | | | |
|----------------------|---|--------------------|--|----------------|---|----------------|--|----------------|---|------------------|---|
| NAME | addchstr, addchnstr, mvaddchstr, mvaddchnstr, mvwaddchnstr, mvwaddchstr, waddchstr, waddchnstr – copy a character string (with renditions) to a window | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addchstr(const chtype *chstr); int addchnstr(const chtype *chstr, int n); int mvaddchnstr(int y, int x, const chtype *chstr, int n); int mvaddchstr(int y, int x, const chtype *chstr); int mvwaddchnstr(WINDOW *win, int y, int x, const chtype *chstr, int n); int mvwaddchstr(WINDOW *win, int y, int x, const chtype *chstr); int waddchstr(WINDOW *win, const chtype *chstr); int waddchnstr(WINDOW *win, const chtype *chstr, int n);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The <code>addchstr()</code> function copies the <code>chtype</code> character string to the <code>stdscr</code> window at the current cursor position. The <code>mvaddchstr()</code> and <code>mvwaddchstr()</code> functions copy the character string to the starting position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <code>win</code>). The <code>waddchstr()</code> is identical to <code>addchstr()</code>, but writes to the window specified by <code>win</code>.</p> <p>The <code>addchnstr()</code>, <code>waddchnstr()</code>, <code>mvaddchnstr()</code>, and <code>mvwaddchnstr()</code> functions write <code>n</code> characters to the window, or as many as will fit on the line. If <code>n</code> is less than 0, the entire string is written, or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <code>x</code> and <code>y</code> parameters.</p> <p>These functions differ from the <code>addstr(3XCURSES)</code> set of functions in two important respects. First, these functions do <i>not</i> advance the cursor after writing the string to the window. Second, the current window rendition is not combined with the character; only the attributes that are already part of the <code>chtype</code> character are used.</p> | | | | | | | | | | |
| PARAMETERS | <table><tr><td><code>chstr</code></td><td>Is a pointer to the <code>chtype</code> string to be copied to the window.</td></tr><tr><td><code>n</code></td><td>Is the maximum number of characters to be copied from <code>chstr</code>. If <code>n</code> is less than 0, the entire string is written or as much of it as fits on the line.</td></tr><tr><td><code>y</code></td><td>Is the <code>y</code> (row) coordinate of the starting position of <code>chstr</code> in the window.</td></tr><tr><td><code>x</code></td><td>Is the <code>x</code> (column) coordinate of the starting position of <code>chstr</code> in the window.</td></tr><tr><td><code>win</code></td><td>Is a pointer to the window to which the string is to be copied.</td></tr></table> | <code>chstr</code> | Is a pointer to the <code>chtype</code> string to be copied to the window. | <code>n</code> | Is the maximum number of characters to be copied from <code>chstr</code> . If <code>n</code> is less than 0, the entire string is written or as much of it as fits on the line. | <code>y</code> | Is the <code>y</code> (row) coordinate of the starting position of <code>chstr</code> in the window. | <code>x</code> | Is the <code>x</code> (column) coordinate of the starting position of <code>chstr</code> in the window. | <code>win</code> | Is a pointer to the window to which the string is to be copied. |
| <code>chstr</code> | Is a pointer to the <code>chtype</code> string to be copied to the window. | | | | | | | | | | |
| <code>n</code> | Is the maximum number of characters to be copied from <code>chstr</code> . If <code>n</code> is less than 0, the entire string is written or as much of it as fits on the line. | | | | | | | | | | |
| <code>y</code> | Is the <code>y</code> (row) coordinate of the starting position of <code>chstr</code> in the window. | | | | | | | | | | |
| <code>x</code> | Is the <code>x</code> (column) coordinate of the starting position of <code>chstr</code> in the window. | | | | | | | | | | |
| <code>win</code> | Is a pointer to the window to which the string is to be copied. | | | | | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | | | | | |

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addch(3XCURSES)`, `addnstr(3XCURSES)`, `attroff(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

mvaddnstr(3CURSES)

NAME | curs_addstr, addstr, addnstr, waddstr, waddnstr, mvaddstr, mvaddnstr, mvwaddstr, mvwaddnstr – add a string of characters to a curses window and advance cursor

SYNOPSIS |

```
cc [ flag ... ] file ... -lcurses [ library ... ]
#include <curses.h>

int addstr(char *str);
int addnstr(char *str, int n);
int waddstr(WINDOW *win, char *str);
int waddnstr(WINDOW *win, char *str, int n);
int mvaddstr(int y, int x, char *str);
int mvaddnstr(int y, int x, char *str, int n);
int mvwaddstr(WINDOW *win, int y, int x, char *str);
int mvwaddnstr(WINDOW *win, int y, int x, char *str, int n);
```

DESCRIPTION | All of these routines write all the characters of the null terminated character string *str* on the given window. It is similar to calling `waddch()` once for each character in the string. The four routines with *n* as the last argument write at most *n* characters. If *n* is negative, then the entire string will be added.

RETURN VALUES | All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | `curs_addch(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES | The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that all routines except `waddstr()` and `waddnstr()` may not be macros.

| | |
|--------------------|---|
| NAME | addnstr, addstr, mvaddnstr, mvaddstr, mvwaddnstr, mvwaddstr, waddnstr, waddstr – add a multi-byte character string (without rendition) to a window |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addnstr(const char *str, int n); int addstr(const char *str); int mvaddnstr(int y, int x, const char *str, int n); int mvaddstr(int y, int x, const char *str); int mvwaddnstr(WINDOW *win, int y, int x, const char *str, int n); int mvwaddstr(WINDOW *win, int y, int x, const char *str); int waddstr(WINDOW *win, const char *str); int waddnstr(WINDOW *win, const char *str, int n);</pre> |
| DESCRIPTION | <p>The <code>addstr()</code> function writes a null-terminated string of multi-byte characters to the <code>stdscr</code> window at the current cursor position. The <code>waddstr()</code> function performs an identical action, but writes the character to the window specified by <i>win</i>. The <code>mvaddstr()</code> and <code>mvwaddstr()</code> functions write the string to the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <i>win</i>).</p> <p>The <code>addnstr()</code>, <code>waddnstr()</code>, <code>mvaddnstr()</code>, and <code>mvwaddnstr()</code> functions are similar but write at most <i>n</i> characters to the window. If <i>n</i> is less than 0, the entire string is written.</p> <p>All of these functions advance the cursor after writing the string.</p> <p>These functions are functionally equivalent to calling the corresponding function from the <code>addch(3XCURSES)</code> set of functions once for each character in the string. Refer to the <code>curses(3XCURSES)</code> man page for a complete description of special character handling and of the interaction between the window rendition (or background character and rendition) and the character written.</p> <p>Note that these functions differ from the <code>addchstr()</code> set of functions in that the <code>addchstr(3XCURSES)</code> functions copy the string as is (without combining each character with the window rendition or the background character and rendition).</p> |
| PARAMETERS | <p><i>str</i> Is a pointer to the character string that is to be written to the window.</p> <p><i>n</i> Is the maximum number of characters to be copied from <i>str</i>. If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line.</p> <p><i>y</i> Is the <i>y</i> (row) coordinate of the starting position of <i>str</i> in the window.</p> |

`mvaddnstr(3XCURSES)`

x Is the x (column) coordinate of the starting position of *str* in the window.

win Is a pointer to the window in which the string is to be written.

RETURN VALUES On success, these functions return `OK`. Otherwise, they return `ERR`.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addch(3XCURSES)`, `addchstr(3XCURSES)`, `curses(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_addwstr, addwstr, addnwstr, waddwstr, waddnwstr, mvaddwstr, mvaddnwstr, mvwaddwstr, mvwaddnwstr – add a string of wchar_t characters to a curses window and advance cursor | | | | |
|---------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lcurses [library...] #include<curses.h> int addwstr(wchar_t *wstr); int addnwstr(wchar_t *wstr, int n); int waddwstr(WINDOW *win, wchar_t *wstr); int waddnwstr(WINDOW *win, wchar_t *wstr, int n); int mvaddwstr(int y, int x, wchar_t *wstr); int mvaddnwstr(int y, int x, wchar_t *wstr, int n); int mvwaddwstr(WINDOW *win, int y, int x, wchar_t *wstr); int mvwaddnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);</pre> | | | | |
| DESCRIPTION | All of these routines write all the characters of the null-terminated wchar_t character string wstr on the given window. The effect is similar to calling waddwch(3CURSES) once for each wchar_t character in the string. The four routines with n as the last argument write at most n wchar_t characters. If n is negative, then the entire string will be added. | | | | |
| RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), waddwch(3CURSES), attributes(5) | | | | |
| NOTES | <p>The header file <curses.h> automatically includes the header files <stdio.h>, <nctrl.h> and <widec.h>.</p> <p>Note that all of these routines except waddwstr() and waddnwstr() may be macros.</p> | | | | |

mvaddnwstr(3XCURSES)

| | |
|--------------------|---|
| NAME | <code>addnwstr</code> , <code>addwstr</code> , <code>mvaddnwstr</code> , <code>mvaddwstr</code> , <code>mvwaddnwstr</code> , <code>mvwaddwstr</code> , <code>waddnwstr</code> , <code>waddwstr</code> – add a wide-character string to a window |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addnwstr(const wchar_t *wstr, int n); int addwstr(const wchar_t *wstr); int mvaddnwstr(int y, int x, const wchar_t *wstr, int n); int mvaddwstr(int y, int x, const wchar_t *wstr); int mvwaddnwstr(WINDOW*win, int y, int x, const wchar_t *wstr, int n); int mvwaddwstr(WINDOW*win, int y, int x, const wchar_t *wstr); int waddnwstr(WINDOW*win, const wchar_t *wstr, int n); int waddwstr(WINDOW*win, const wchar_t *wstr);</pre> |
| DESCRIPTION | <p>The <code>addwstr()</code> function writes a null-terminated wide-character string to the <code>stdscr</code> window at the current cursor position. The <code>waddwstr()</code> function performs an identical action, but writes the string to the window specified by <code>win</code>. The <code>mvaddwstr()</code> and <code>mvwaddwstr()</code> functions write the string to the position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <code>win</code>).</p> <p>The <code>addnwstr()</code>, <code>waddnwstr()</code>, <code>mvaddnwstr()</code>, and <code>mvwaddnwstr()</code> functions write at most <code>n</code> characters to the window. If <code>n</code> is less than 0, the entire string is written. The former two functions place the characters at the current cursor position; the latter two commands use the position specified by the <code>x</code> and <code>y</code> parameters.</p> <p>All of these functions advance the cursor after writing the string.</p> <p>These functions are functionally equivalent to building a <code>cchar_t</code> from the <code>wchar_t</code> and the window rendition (or background character and rendition) and calling the <code>wadd_wch(3XCURSES)</code> function once for each <code>wchar_t</code> in the string. Refer to the <code>curses(3XCURSES)</code> man page for a complete description of special character handling and of the interaction between the window rendition (or background character and rendition) and the character written.</p> <p>Note that these functions differ from the <code>add_wchnstr(3XCURSES)</code> set of functions in that the latter copy the string as is (without combining each character with the foreground and background attributes of the window).</p> |
| PARAMETERS | <p><code>wstr</code> Is a pointer to the wide-character string that is to be written to the window.</p> |

mvaddnwstr(3XCURSES)

n Is the maximum number of characters to be copied from *wstr*. If *n* is less than 0, the entire string is written or as much of it as fits on the line.

y Is the y (row) coordinate of the starting position of *wstr* in the window.

x Is the x (column) coordinate of the starting position of *wstr* in the window.

win Is a pointer to the window in which the string is to be written.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `add_wch(3XCURSES)`, `add_wchnstr(3XCURSES)`, `curses(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

mvaddstr(3CURSES)

NAME | curs_addstr, addstr, addnstr, waddstr, waddnstr, mvaddstr, mvaddnstr, mvwaddstr, mvwaddnstr – add a string of characters to a curses window and advance cursor

SYNOPSIS | `cc [flag ...] file ... -lcurses [library ...]
#include <curses.h>`

```
int addstr(char *str);  
int addnstr(char *str, int n);  
int waddstr(WINDOW *win, char *str);  
int waddnstr(WINDOW *win, char *str, int n);  
int mvaddstr(int y, int x, char *str);  
int mvaddnstr(int y, int x, char *str, int n);  
int mvwaddstr(WINDOW *win, int y, int x, char *str);  
int mvwaddnstr(WINDOW *win, int y, int x, char *str, int n);
```

DESCRIPTION | All of these routines write all the characters of the null terminated character string *str* on the given window. It is similar to calling `waddch()` once for each character in the string. The four routines with *n* as the last argument write at most *n* characters. If *n* is negative, then the entire string will be added.

RETURN VALUES | All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | `curs_addch(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES | The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that all routines except `waddstr()` and `waddnstr()` may not be macros.

| | |
|--------------------|---|
| NAME | addnstr, addstr, mvaddnstr, mvaddstr, mvwaddnstr, mvwaddstr, waddnstr, waddstr – add a multi-byte character string (without rendition) to a window |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addnstr(const char *str, int n); int addstr(const char *str); int mvaddnstr(int y, int x, const char *str, int n); int mvaddstr(int y, int x, const char *str); int mvwaddnstr(WINDOW *win, int y, int x, const char *str, int n); int mvwaddstr(WINDOW *win, int y, int x, const char *str); int waddstr(WINDOW *win, const char *str); int waddnstr(WINDOW *win, const char *str, int n);</pre> |
| DESCRIPTION | <p>The <code>addstr()</code> function writes a null-terminated string of multi-byte characters to the <code>stdscr</code> window at the current cursor position. The <code>waddstr()</code> function performs an identical action, but writes the character to the window specified by <i>win</i>. The <code>mvaddstr()</code> and <code>mvwaddstr()</code> functions write the string to the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <i>win</i>).</p> <p>The <code>addnstr()</code>, <code>waddnstr()</code>, <code>mvaddnstr()</code>, and <code>mvwaddnstr()</code> functions are similar but write at most <i>n</i> characters to the window. If <i>n</i> is less than 0, the entire string is written.</p> <p>All of these functions advance the cursor after writing the string.</p> <p>These functions are functionally equivalent to calling the corresponding function from the <code>addch(3XCURSES)</code> set of functions once for each character in the string. Refer to the <code>curses(3XCURSES)</code> man page for a complete description of special character handling and of the interaction between the window rendition (or background character and rendition) and the character written.</p> <p>Note that these functions differ from the <code>addchstr()</code> set of functions in that the <code>addchstr(3XCURSES)</code> functions copy the string as is (without combining each character with the window rendition or the background character and rendition).</p> |
| PARAMETERS | <p><i>str</i> Is a pointer to the character string that is to be written to the window.</p> <p><i>n</i> Is the maximum number of characters to be copied from <i>str</i>. If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line.</p> <p><i>y</i> Is the <i>y</i> (row) coordinate of the starting position of <i>str</i> in the window.</p> |

mvaddstr(3XCURSES)

x Is the x (column) coordinate of the starting position of *str* in the window.

win Is a pointer to the window in which the string is to be written.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addch(3XCURSES)`, `addchstr(3XCURSES)`, `curses(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|----------------------|---|
| NAME | curs_addwch, addwch, waddwch, mvaddwch, mvwaddwch, echowchar, wechowchar – add a wchar_t character (with attributes) to a curses window and advance cursor |
| SYNOPSIS | <pre>cc [flag...] file... -lcurses [library...] #include<curses.h> int addwch (chtype wch) ; int waddwch (WINDOW *win, chtype wch) ; int mvaddwch (int y, int x, chtype wch) ; int mvwaddwch (WINDOW *win, int y, int x, chtype wch) ; int echowchar (chtype wch) ; int wechowchar (WINDOW *win, chtype wch) ;</pre> |
| DESCRIPTION | <p>The addwch(), waddwch(), mvaddwch(), and mvwaddwch() routines put the character <i>wch</i>, holding a wchar_t character, into the window at the current cursor position of the window and advance the position of the window cursor. Their function is similar to that of putwchar(3C) in the C multibyte library. At the right margin, an automatic newline is performed. At the bottom of the scrolling region, if scrolllok is enabled, the scrolling region is scrolled up one line.</p> <p>If <i>wch</i> is a tab, newline, or backspace, the cursor is moved appropriately within the window. A newline also does a clrtoeol(3CURSES) before moving. Tabs are considered to be at every eighth column. If <i>wch</i> is another control character, it is drawn in the ^X notation. Calling winwch(3CURSES) after adding a control character does not return the control character, but instead returns the representation of the control character.</p> <p>Video attributes can be combined with a wchar_t character by OR-ing them into the parameter. This results in these attributes also being set. (The intent here is that text, including attributes, can be copied from one place to another using inwch() and addwch().) See standout(3CURSES), predefined video attribute constants.</p> <p>The echowchar() and wechowchar() routines are functionally equivalent to a call to addwch() followed by a call to refresh(3CURSES), or a call to waddwch() followed by a call to wrefresh(3CURSES). The knowledge that only a single character is being output is taken into consideration and, for non-control characters, a considerable performance gain might be seen by using these routines instead of their equivalents.</p> |
| Line Graphics | <p>The following variables may be used to add line drawing characters to the screen with routines of the addwch() family. When variables are defined for the terminal, the A_ALTCHARSET bit is turned on. (See curs_attr(3CURSES)). Otherwise, the default character listed below is stored in the variable. The names chosen are consistent with the VT100 nomenclature.</p> |

mvaddwch(3CURSES)

| Name | Default | Glyph Description |
|--------------|---------|-------------------------|
| ACS_ULCORNER | + | upper left-hand corner |
| ACS_LLCORNER | + | lower left-hand corner |
| ACS_URCORNER | + | upper right-hand corner |
| ACS_LRCORNER | + | lower right-hand corner |
| ACS_RTEE | + | right tee |
| ACS_LTEE | + | left tee |
| ACS_BTEE | + | bottom tee |
| ACS_TTEE | + | top tee |
| ACS_HLINE | - | horizontal line |
| ACS_VLINE | | vertical line |
| ACS_PLUS | + | plus |
| ACS_S1 | - | scan line 1 |
| ACS_S9 | - | scan line 9 |
| ACS_DIAMOND | + | diamond |
| ACS_CKBOARD | : | checker board (stipple) |
| ACS_DEGREE | ' | degree symbol |
| ACS_PLMINUS | # | plus/minus |
| ACS_BULLET | o | bullet |
| ACS_LARROW | < | arrow pointing left |
| ACS_RARRROW | > | arrow pointing right |
| ACS_DARROW | v | arrow pointing down |
| ACS_UARROW | ^ | arrow pointing up |
| ACS_BOARD | # | board of squares |
| ACS_LANTERN | # | lantern symbol |
| ACS_BLOCK | # | solid square block |

RETURN VALUE

All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion, unless otherwise noted in the preceding routine descriptions.

mvaddwch(3CURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `putwchar(3C)`, `clrtoeol(3CURSES)`, `curses(3CURSES)`, `curl_attr(3CURSES)`, `curl_inwch(3CURSES)`, `curl_outopts(3CURSES)`, `refresh(3CURSES)`, `standout(3CURSES)`, `winwch(3CURSES)`, `wrefresh(3CURSES)`, `attributes(5)`

NOTES The header file `<curses.h>` automatically includes the header files `<stdio.h>`, `<unctrl.h>` and `<widec.h>`.

Note that `addwch()`, `mvaddwch()`, `mvwaddwch()`, and `echowchar()` may be macros.

None of these routines can use the color attribute in `chtype`.

mvadd_wch(3XCURSES)

| | |
|--------------------|--|
| NAME | add_wch, mvadd_wch, mvwadd_wch, wadd_wch – add a complex character (with rendition) to a window |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int add_wch(const cchar_t *wch); int wadd_wch(WINDOW *win, const cchar_t *wch); int mvadd_wch(int y, int x, const cchar_t *wch); int mvwadd_wch(WINDOW *win, int y, int x, const cchar_t *wch);</pre> |
| DESCRIPTION | <p>The <code>add_wch()</code> function writes a complex character to the <code>stdscr</code> window at the current cursor position. The <code>mvadd_wch()</code> and <code>mvwadd_wch()</code> functions write the character to the position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters. The <code>mvadd_wch()</code> function writes the character to the <code>stdscr</code> window, while <code>mvwadd_wch()</code> writes the character to the window specified by <code>win</code>. The <code>wadd_wch()</code> function is identical to <code>add_wch()</code>, but writes the character to the window specified by <code>win</code>. These functions advance the cursor after writing the character.</p> <p>If <code>wch</code> is a spacing complex character, X/Open Curses replaces any previous character at the specified location with <code>wch</code> (and its rendition). If <code>wch</code> is a non-spacing complex character, X/Open Curses preserves all existing characters at the specified location and adds the non-spacing characters of <code>wch</code> to the spacing complex character. It ignores the rendition associated with <code>wch</code>.</p> <p>Characters that do not fit on the end of the current line are wrapped to the beginning of the next line unless the current line is the last line of the window and scrolling is disabled. In that situation, X/Open Curses discards characters which extend beyond the end of the line.</p> <p>When <code>wch</code> is a backspace, carriage return, newline, or tab, X/Open Curses moves the cursor appropriately as described in the <code>curses(3XCURSES)</code> man page. Each tab character moves the cursor to the next tab stop. By default, tab stops occur every eight columns. When <code>wch</code> is a control character other than a backspace, carriage return, newline, or tab, it is written using <code>^x</code> notation, where <code>x</code> is a printable character. When X/Open Curses writes <code>wch</code> to the last character position on a line, it automatically generates a newline. When <code>wch</code> is written to the last character position of a scrolling region and <code>scrollok()</code> is enabled, X/Open Curses scrolls the scrolling region up one line (see <code>clearok(3XCURSES)</code>).</p> |
| PARAMETERS | <p><code>wch</code> Is the character/attribute pair (rendition) to be written to the window.</p> <p><code>win</code> Is a pointer to the window in which the character is to be written.</p> <p><code>y</code> Is the <code>y</code> (row) coordinate of the character's position in the window.</p> |

mvadd_wch(3XCURSES)

x Is the *x* (column) coordinate of the character's position in the window.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attr_off(3XCURSES)`, `bkgrndset(3XCURSES)`, `curses(3XCURSES)`, `doupdate(3XCURSES)`, `in_wch(3XCURSES)`, `ins_wch(3XCURSES)`, `libcurses(3XCURSES)`, `n1(3XCURSES)`, `printw(3XCURSES)`, `scrollok(3XCURSES)`, `scl1(3XCURSES)`, `setscrreg(3XCURSES)`, `terminfo(4)`, `attributes(5)`, `standards(5)`

mvaddwchnstr(3CURSES)

| NAME | <p> <code> curs_addwchstr, addwchstr, addwchnstr, waddwchstr, waddwchnstr, mvaddwchstr, mvaddwchnstr, mvwaddwchstr, mvwaddwchnstr</code> – add string of <code>wchar_t</code> characters (and attributes) to a curses window </p> | | | | |
|---------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre> cc [flag...] file... -lcurses [library...] #include<curses.h> int addwchstr(chtype *wchstr); int addwchnstr(chtype *wchstr, int n); int waddwchstr(WINDOW *win, chtype *wchstr); int waddwchnstr(WINDOW *win, chtype *wchstr, int n); int mvaddwchstr(int y, int x, chtype *wchstr); int mvaddwchnstr(int y, int x, chtype *wchstr, int n); int mvwaddwchstr(WINDOW *win, int y, int x, chtype *wchstr); int mvwaddwchnstr(WINDOW *win, int y, int x, chtype *wchstr, int n); </pre> | | | | |
| DESCRIPTION | <p> All of these routines copy <code>wchstr</code>, which points to a string of <code>wchar_t</code> characters, directly into the window image structure starting at the current cursor position. The four routines with <code>n</code> as the last argument copy at most <code>n</code> elements, but no more than will fit on the line. If <code>n=-1</code> then the whole string is copied, to the maximum number that fit on the line. </p> <p> The position of the window cursor is not advanced. These routines work faster than <code>waddnwstr(3CURSES)</code> because they merely copy <code>wchstr</code> into the window image structure. On the other hand, care must be taken when using these functions because they don't perform any kind of checking (such as for the newline character), they do not advance the current cursor position, and they truncate the string, rather than wrapping it around to the new line. </p> | | | | |
| RETURN VALUE | <p> All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion, unless otherwise noted in the preceding routine descriptions. </p> | | | | |
| ATTRIBUTES | <p> See <code>attributes(5)</code> for descriptions of the following attributes: </p> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">MT-Level</td> <td style="text-align: center;">Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <p> <code>curses(3CURSES)</code>, <code>waddnwstr(3CURSES)</code>, <code>attributes(5)</code> </p> | | | | |
| NOTES | <p> The header file <code><curses.h></code> automatically includes the header files <code><stdio.h></code>, <code><unctrl.h></code> and <code><widec.h></code>. </p> | | | | |

`mvaddwchnstr(3CURSES)`

Note that all routines except `waddwchnstr()` may be macros.

None of these routines can use the color attribute in `chtype`.

mvadd_wchnstr(3XCURSES)

| | | | | | | | |
|---------------------|--|---------------------|---|----------------|--|----------------|---|
| NAME | add_wchnstr, add_wchstr, mvadd_wchnstr, mvadd_wchstr, mvwadd_wchnstr, mvwadd_wchstr, wadd_wchnstr, wadd_wchstr – copy a string of complex characters (with renditions) to a window | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int add_wchnstr(const cchar_t *wchstr, int n); int add_wchstr(const cchar_t *wchstr); int mvadd_wchnstr(int y, int x, const cchar_t *wchstr, int n); int mvadd_wchstr(int y, int x, const cchar_t *wchstr); int mvwadd_wchnstr(WINDOW *win, int y, int x, const cchar_t *wchstr, int n); int mvwadd_wchstr(WINDOW *win, int y, int x, const cchar_t *wchstr); int wadd_wchstr(WINDOW *win, const cchar_t *wchstr); int wadd_wchnstr(WINDOW *win, const cchar_t *wchstr, int n);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>add_wchstr()</code> function copies the string of <code>cchar_t</code> characters to the <code>stdscr</code> window at the current cursor position. The <code>mvadd_wchstr()</code> and <code>mvwadd_wchstr()</code> functions copy the string to the starting position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <code>win</code>). The <code>wadd_wchstr()</code> is identical to <code>add_wchstr()</code>, but writes to the window specified by <code>win</code>.</p> <p>The <code>add_wchnstr()</code>, <code>wadd_wchnstr()</code>, <code>mvadd_wchnstr()</code>, and <code>mvwadd_wchnstr()</code> functions write <code>n</code> characters to the window, or as many as will fit on the line. If <code>n</code> is less than 0, the entire string is written, or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <code>x</code> and <code>y</code> parameters.</p> <p>These functions differ from the <code>addwstr(3XCURSES)</code> set of functions in two important respects. First, these functions do <i>not</i> advance the cursor after writing the string to the window. Second, the current window rendition (that is, the combination of attributes and color pair) is not combined with the character; only those attributes that are already part of the <code>cchar_t</code> character are used.</p> | | | | | | |
| PARAMETERS | <table><tr><td><code>wchstr</code></td><td>Is a pointer to the <code>cchar_t</code> string to be copied to the window.</td></tr><tr><td><code>n</code></td><td>Is the maximum number of characters to be copied from <code>wchstr</code>. If <code>n</code> is less than 0, the entire string is written or as much of it as fits on the line.</td></tr><tr><td><code>y</code></td><td>Is the <code>y</code> (row) coordinate of the starting position of <code>wchstr</code> in the window.</td></tr></table> | <code>wchstr</code> | Is a pointer to the <code>cchar_t</code> string to be copied to the window. | <code>n</code> | Is the maximum number of characters to be copied from <code>wchstr</code> . If <code>n</code> is less than 0, the entire string is written or as much of it as fits on the line. | <code>y</code> | Is the <code>y</code> (row) coordinate of the starting position of <code>wchstr</code> in the window. |
| <code>wchstr</code> | Is a pointer to the <code>cchar_t</code> string to be copied to the window. | | | | | | |
| <code>n</code> | Is the maximum number of characters to be copied from <code>wchstr</code> . If <code>n</code> is less than 0, the entire string is written or as much of it as fits on the line. | | | | | | |
| <code>y</code> | Is the <code>y</code> (row) coordinate of the starting position of <code>wchstr</code> in the window. | | | | | | |

mvadd_wchnstr(3XCURSES)

x Is the *x* (column) coordinate of the starting position of *wchstr* in the window.

win Is a pointer to the window to which the string is to be copied.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addnwstr(3XCURSES)`, `add_wch(3XCURSES)`, `attr_off(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

mvaddwchstr(3CURSES)

| NAME | <p> <code> curs_addwchstr, addwchstr, addwchnstr, waddwchstr, waddwchnstr, mvaddwchstr, mvaddwchnstr, mvwaddwchstr, mvwaddwchnstr</code> – add string of <code>wchar_t</code> characters (and attributes) to a curses window </p> | | | | |
|---------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre> cc [flag...] file... -lcurses [library...] #include<curses.h> int addwchstr(chtype *wchstr); int addwchnstr(chtype *wchstr, int n); int waddwchstr(WINDOW *win, chtype *wchstr); int waddwchnstr(WINDOW *win, chtype *wchstr, int n); int mvaddwchstr(int y, int x, chtype *wchstr); int mvaddwchnstr(int y, int x, chtype *wchstr, int n); int mvwaddwchstr(WINDOW *win, int y, int x, chtype *wchstr); int mvwaddwchnstr(WINDOW *win, int y, int x, chtype *wchstr, int n); </pre> | | | | |
| DESCRIPTION | <p> All of these routines copy <code>wchstr</code>, which points to a string of <code>wchar_t</code> characters, directly into the window image structure starting at the current cursor position. The four routines with <code>n</code> as the last argument copy at most <code>n</code> elements, but no more than will fit on the line. If <code>n=-1</code> then the whole string is copied, to the maximum number that fit on the line. </p> <p> The position of the window cursor is not advanced. These routines work faster than <code>waddnwstr(3CURSES)</code> because they merely copy <code>wchstr</code> into the window image structure. On the other hand, care must be taken when using these functions because they don't perform any kind of checking (such as for the newline character), they do not advance the current cursor position, and they truncate the string, rather than wrapping it around to the new line. </p> | | | | |
| RETURN VALUE | <p> All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion, unless otherwise noted in the preceding routine descriptions. </p> | | | | |
| ATTRIBUTES | <p> See <code>attributes(5)</code> for descriptions of the following attributes: </p> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">MT-Level</td> <td style="text-align: center;">Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <p> <code>curses(3CURSES)</code>, <code>waddnwstr(3CURSES)</code>, <code>attributes(5)</code> </p> | | | | |
| NOTES | <p> The header file <code><curses.h></code> automatically includes the header files <code><stdio.h></code>, <code><unctrl.h></code> and <code><widec.h></code>. </p> | | | | |

`mvaddwchstr(3CURSES)`

Note that all routines except `waddwchnstr()` may be macros.

None of these routines can use the color attribute in `chtype`.

mvadd_wchstr(3XCURSES)

| | | | | | | | |
|---------------------|--|---------------------|---|----------------|--|----------------|---|
| NAME | add_wchnstr, add_wchstr, mvadd_wchnstr, mvadd_wchstr, mvwadd_wchnstr, mvwadd_wchstr, wadd_wchnstr, wadd_wchstr – copy a string of complex characters (with renditions) to a window | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int add_wchnstr(const cchar_t *wchstr, int n); int add_wchstr(const cchar_t *wchstr); int mvadd_wchnstr(int y, int x, const cchar_t *wchstr, int n); int mvadd_wchstr(int y, int x, const cchar_t *wchstr); int mvwadd_wchnstr(WINDOW *win, int y, int x, const cchar_t *wchstr, int n); int mvwadd_wchstr(WINDOW *win, int y, int x, const cchar_t *wchstr); int wadd_wchstr(WINDOW *win, const cchar_t *wchstr); int wadd_wchnstr(WINDOW *win, const cchar_t *wchstr, int n);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>add_wchstr()</code> function copies the string of <code>cchar_t</code> characters to the <code>stdscr</code> window at the current cursor position. The <code>mvadd_wchstr()</code> and <code>mvwadd_wchstr()</code> functions copy the string to the starting position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <code>win</code>). The <code>wadd_wchstr()</code> is identical to <code>add_wchstr()</code>, but writes to the window specified by <code>win</code>.</p> <p>The <code>add_wchnstr()</code>, <code>wadd_wchnstr()</code>, <code>mvadd_wchnstr()</code>, and <code>mvwadd_wchnstr()</code> functions write <code>n</code> characters to the window, or as many as will fit on the line. If <code>n</code> is less than 0, the entire string is written, or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <code>x</code> and <code>y</code> parameters.</p> <p>These functions differ from the <code>addwstr(3XCURSES)</code> set of functions in two important respects. First, these functions do <i>not</i> advance the cursor after writing the string to the window. Second, the current window rendition (that is, the combination of attributes and color pair) is not combined with the character; only those attributes that are already part of the <code>cchar_t</code> character are used.</p> | | | | | | |
| PARAMETERS | <table><tr><td><code>wchstr</code></td><td>Is a pointer to the <code>cchar_t</code> string to be copied to the window.</td></tr><tr><td><code>n</code></td><td>Is the maximum number of characters to be copied from <code>wchstr</code>. If <code>n</code> is less than 0, the entire string is written or as much of it as fits on the line.</td></tr><tr><td><code>y</code></td><td>Is the <code>y</code> (row) coordinate of the starting position of <code>wchstr</code> in the window.</td></tr></table> | <code>wchstr</code> | Is a pointer to the <code>cchar_t</code> string to be copied to the window. | <code>n</code> | Is the maximum number of characters to be copied from <code>wchstr</code> . If <code>n</code> is less than 0, the entire string is written or as much of it as fits on the line. | <code>y</code> | Is the <code>y</code> (row) coordinate of the starting position of <code>wchstr</code> in the window. |
| <code>wchstr</code> | Is a pointer to the <code>cchar_t</code> string to be copied to the window. | | | | | | |
| <code>n</code> | Is the maximum number of characters to be copied from <code>wchstr</code> . If <code>n</code> is less than 0, the entire string is written or as much of it as fits on the line. | | | | | | |
| <code>y</code> | Is the <code>y</code> (row) coordinate of the starting position of <code>wchstr</code> in the window. | | | | | | |

mvadd_wchstr(3XCURSES)

x Is the *x* (column) coordinate of the starting position of *wchstr* in the window.

win Is a pointer to the window to which the string is to be copied.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addnwstr(3XCURSES)`, `add_wch(3XCURSES)`, `attr_off(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

mvaddwstr(3CURSES)

NAME | curs_addwstr, addwstr, addnwstr, waddwstr, waddnwstr, mvaddwstr, mvaddnwstr, mvwaddwstr, mvwaddnwstr – add a string of wchar_t characters to a curses window and advance cursor

SYNOPSIS | **cc** [flag...] file... -lcurses [library...]

```
#include<curses.h>
int addwstr(wchar_t *wstr);
int addnwstr(wchar_t *wstr, int n);
int waddwstr(WINDOW *win, wchar_t *wstr);
int waddnwstr(WINDOW *win, wchar_t *wstr, int n);
int mvaddwstr(int y, int x, wchar_t *wstr);
int mvaddnwstr(int y, int x, wchar_t *wstr, int n);
int mvwaddwstr(WINDOW *win, int y, int x, wchar_t *wstr);
int mvwaddnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);
```

DESCRIPTION | All of these routines write all the characters of the null-terminated wchar_t character string wstr on the given window. The effect is similar to calling waddwch(3CURSES) once for each wchar_t character in the string. The four routines with n as the last argument write at most n wchar_t characters. If n is negative, then the entire string will be added.

RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | curses(3CURSES), waddwch(3CURSES), attributes(5)

NOTES | The header file <curses.h> automatically includes the header files <stdio.h>, <nctrl.h> and <widec.h>.

Note that all of these routines except waddwstr() and waddnwstr() may be macros.

| | |
|--------------------|---|
| NAME | addnwstr, addwstr, mvaddnwstr, mvaddwstr, mvwaddnwstr, mvwaddwstr, waddnwstr, waddwstr – add a wide-character string to a window |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addnwstr(const wchar_t *wstr, int n); int addwstr(const wchar_t *wstr); int mvaddnwstr(int y, int x, const wchar_t *wstr, int n); int mvaddwstr(int y, int x, const wchar_t *wstr); int mvwaddnwstr(WINDOW*win, int y, int x, const wchar_t *wstr, int n); int mvwaddwstr(WINDOW*win, int y, int x, const wchar_t *wstr); int waddnwstr(WINDOW*win, const wchar_t *wstr, int n); int waddwstr(WINDOW*win, const wchar_t *wstr);</pre> |
| DESCRIPTION | <p>The <code>addwstr()</code> function writes a null-terminated wide-character string to the <code>stdscr</code> window at the current cursor position. The <code>waddwstr()</code> function performs an identical action, but writes the string to the window specified by <code>win</code>. The <code>mvaddwstr()</code> and <code>mvwaddwstr()</code> functions write the string to the position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <code>win</code>).</p> <p>The <code>addnwstr()</code>, <code>waddnwstr()</code>, <code>mvaddnwstr()</code>, and <code>mvwaddnwstr()</code> functions write at most <code>n</code> characters to the window. If <code>n</code> is less than 0, the entire string is written. The former two functions place the characters at the current cursor position; the latter two commands use the position specified by the <code>x</code> and <code>y</code> parameters.</p> <p>All of these functions advance the cursor after writing the string.</p> <p>These functions are functionally equivalent to building a <code>cchar_t</code> from the <code>wchar_t</code> and the window rendition (or background character and rendition) and calling the <code>wadd_wch(3XCURSES)</code> function once for each <code>wchar_t</code> in the string. Refer to the <code>curses(3XCURSES)</code> man page for a complete description of special character handling and of the interaction between the window rendition (or background character and rendition) and the character written.</p> <p>Note that these functions differ from the <code>add_wchnstr(3XCURSES)</code> set of functions in that the latter copy the string as is (without combining each character with the foreground and background attributes of the window).</p> |
| PARAMETERS | <p><code>wstr</code> Is a pointer to the wide-character string that is to be written to the window.</p> |

mvaddwstr(3XCURSES)

- n* Is the maximum number of characters to be copied from *wstr*. If *n* is less than 0, the entire string is written or as much of it as fits on the line.
- y* Is the y (row) coordinate of the starting position of *wstr* in the window.
- x* Is the x (column) coordinate of the starting position of *wstr* in the window.
- win* Is a pointer to the window in which the string is to be written.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `add_wch(3XCURSES)`, `add_wchnstr(3XCURSES)`, `curses(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| | | | | | | | | | | | | | | | |
|----------------------|---|----------|---|-------------|--|--------------|---|-------------|---|----------|---|----------|---|------------|---|
| NAME | chgat, mvchgat, mvwchgat, wchgat – change the rendition of characters in a window | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int chgat(int <i>n</i>, attr_t <i>attr</i>, short <i>color</i>, const void *<i>opts</i>); int mvchgat(int <i>y</i>, int <i>x</i>, int <i>n</i>, attr_t <i>attr</i>, short <i>color</i>, const void *<i>opts</i>); int mvwchgat(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>, int <i>n</i>, attr_t <i>attr</i>, short <i>color</i>, const void *<i>opts</i>); int wchgat(WINDOW *<i>win</i>, int <i>n</i>, attr_t <i>attr</i>, short <i>color</i>, const void *<i>opts</i>);</pre> | | | | | | | | | | | | | | |
| DESCRIPTION | <p>These functions change the renditions of the next <i>n</i> characters in the current or specified window (or of the remaining characters on the current or specified line, if <i>n</i> is -1), beginning at the current or specified cursor position. The attributes and colors are specified by <i>attr</i> and <i>color</i> as for <code>setcchar(3XCURSES)</code>.</p> <p>These function neither update the cursor nor perform wrapping.</p> <p>A value of <i>n</i> that is greater than the remaining characters on a line is not an error.</p> <p>The <i>opts</i> argument is reserved for definition in a future release. Currently, the application must provide a null pointer for <i>opts</i>.</p> | | | | | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>n</i></td> <td>Is the number of characters whose rendition is to be changed.</td> </tr> <tr> <td><i>attr</i></td> <td>Is the set of attributes to be assigned to the characters.</td> </tr> <tr> <td><i>color</i></td> <td>Is the new color pair to be assigned to the characters.</td> </tr> <tr> <td><i>opts</i></td> <td>Is reserved for future use. Currently, this must be a null pointer.</td> </tr> <tr> <td><i>y</i></td> <td>Is the y (row) coordinate of the starting position in the window.</td> </tr> <tr> <td><i>x</i></td> <td>Is the x (column) coordinate of the starting position in the window. changed in the window.</td> </tr> <tr> <td><i>win</i></td> <td>Is a pointer to the window in which the rendition of characters is to be changed.</td> </tr> </table> | <i>n</i> | Is the number of characters whose rendition is to be changed. | <i>attr</i> | Is the set of attributes to be assigned to the characters. | <i>color</i> | Is the new color pair to be assigned to the characters. | <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | <i>y</i> | Is the y (row) coordinate of the starting position in the window. | <i>x</i> | Is the x (column) coordinate of the starting position in the window. changed in the window. | <i>win</i> | Is a pointer to the window in which the rendition of characters is to be changed. |
| <i>n</i> | Is the number of characters whose rendition is to be changed. | | | | | | | | | | | | | | |
| <i>attr</i> | Is the set of attributes to be assigned to the characters. | | | | | | | | | | | | | | |
| <i>color</i> | Is the new color pair to be assigned to the characters. | | | | | | | | | | | | | | |
| <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | | | | | | | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of the starting position in the window. | | | | | | | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of the starting position in the window. changed in the window. | | | | | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the rendition of characters is to be changed. | | | | | | | | | | | | | | |
| RETURN VALUES | Upon successful completion, these functions returned OK. Otherwise, they return ERR. | | | | | | | | | | | | | | |
| ERRORS | No errors are defined. | | | | | | | | | | | | | | |

mvchgat(3XCURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `bkgrnd(3XCURSES)`, `libcurses(3XCURSES)`, `setcchar(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | curs_terminfo, setupterm, setterm, set_curterm, del_curterm, restartterm, tparm, tputs, putp, vidputs, vidattr, mvcur, tigetflag, tigetnum, tigetstr – curses interfaces to terminfo database |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> #include <term.h> int setupterm(char *term, int fildes, int *errret); int setterm(char *term); int set_curterm(TERMINAL *nterm); int del_curterm(TERMINAL *oterm); int restartterm(char *term, int fildes, int *errret); char *tparm(char *str, long int p1, long int p2, long int p3, long int p4, long int p5, long int p6, long int p7, long int p8, long int p9); int tputs(char *str, int affcnt, int (*putc)(char)); int putp(char *str); int vidputs(chtype attrs, int (*putc)(char)); int vidattr(chtype attrs); int mvcur(int oldrow, int oldcol, int newrow, int newcol); int tigetflag(char *capname); int tigetnum(char *capname); char *tigetstr(char *capname);</pre> |
| DESCRIPTION | <p>These low-level routines must be called by programs that have to deal directly with the <i>terminfo</i> database to handle certain terminal capabilities, such as programming function keys. For all other functionality, curses routines are more suitable and their use is recommended.</p> <p>Initially, <code>setupterm()</code> should be called. Note that <code>setupterm()</code> is automatically called by <code>initscr()</code> and <code>newterm()</code>. This defines the set of terminal-dependent variables (listed in <code>terminfo(4)</code>). The <i>terminfo</i> variables <code>lines</code> and <code>columns</code> are initialized by <code>setupterm()</code> as follows: If <code>use_env(FALSE)</code> has been called, values for <code>lines</code> and <code>columns</code> specified in <i>terminfo</i> are used. Otherwise, if the environment variables <code>LINES</code> and <code>COLUMNS</code> exist, their values are used. If these environment variables do not exist and the program is running in a window, the current window size is used. Otherwise, if the environment variables do not exist, the values for <code>lines</code> and <code>columns</code> specified in the <i>terminfo</i> database are used.</p> <p>The headers <code><curses.h></code> and <code><term.h></code> should be included (in this order) to get the definitions for these strings, numbers, and flags. Parameterized strings should be passed through <code>tparm()</code> to instantiate them. All <i>terminfo</i> strings (including the output</p> |

mvcur(3CURSES)

of `tparm()` should be printed with `tputs()` or `putp()`. Call the `reset_shell_mode()` routine to restore the tty modes before exiting (see `curs_kernel(3CURSES)`). Programs which use cursor addressing should output `enter_ca_mode` upon startup and should output `exit_ca_mode` before exiting. Programs desiring shell escapes should call `reset_shell_mode` and output `exit_ca_mode` before the shell is called and should output `enter_ca_mode` and call `reset_prog_mode` after returning from the shell.

The `setupterm()` routine reads in the *terminfo* database, initializing the *terminfo* structures, but does not set up the output virtualization structures used by *curses*. The terminal type is the character string *term*; if *term* is null, the environment variable `TERM` is used. All output is to file descriptor *fildev* which is initialized for output. If *errret* is not null, then `setupterm()` returns `OK` or `ERR` and stores a status value in the integer pointed to by *errret*. A status of 1 in *errret* is normal, 0 means that the terminal could not be found, and -1 means that the *terminfo* database could not be found. If *errret* is null, `setupterm()` prints an error message upon finding an error and exits. Thus, the simplest call is:

```
setupterm((char *)0, 1, (int *)0);
```

which uses all the defaults and sends the output to `stdout`.

The `setterm()` routine is being replaced by `setupterm()`. The call:

```
setupterm(term, 1, (int *)0)
```

provides the same functionality as `setterm(term)`. The `setterm()` routine is included here for compatibility and is supported at Level 2.

The `set_curterm()` routine sets the variable `cur_term` to *nterm*, and makes all of the *terminfo* boolean, numeric, and string variables use the values from *nterm*.

The `del_curterm()` routine frees the space pointed to by *oterm* and makes it available for further use. If *oterm* is the same as `cur_term`, references to any of the *terminfo* boolean, numeric, and string variables thereafter may refer to invalid memory locations until another `setupterm()` has been called.

The `restartterm()` routine is similar to `setupterm()` and `initscr()`, except that it is called after restoring memory to a previous state. It assumes that the windows and the input and output options are the same as when memory was saved, but the terminal type and baud rate may be different.

The `tparm()` routine instantiates the string *str* with parameters *pi*. A pointer is returned to the result of *str* with the parameters applied.

The `tputs()` routine applies padding information to the string *str* and outputs it. The *str* must be a *terminfo* string variable or the return value from `tparm()`, `tgetstr()`, or `tgoto()`. *affcnt* is the number of lines affected, or 1 if not applicable. *putc* is a `putchar()`-like routine to which the characters are passed, one at a time.

The `putp()` routine calls `tputs(str, 1, putchar)`. Note that the output of `putpA()` always goes to `stdout`, not to the *files* specified in `setupterm()`.

The `vidputs()` routine displays the string on the terminal in the video attribute mode *attrs*, which is any combination of the attributes listed in `curses(3CURSES)`. The characters are passed to the `putchar()`-like routine `putc()`.

The `vidattr()` routine is like the `vidputs()` routine, except that it outputs through `putchar()`.

The `mvcur()` routine provides low-level cursor motion.

The `tigetflag()`, `tigetnum()` and `tigetstr()` routines return the value of the capability corresponding to the *terminfo capname* passed to them, such as `xenl`.

With the `tigetflag()` routine, the value `-1` is returned if *capname* is not a boolean capability.

With the `tigetnum()` routine, the value `-2` is returned if *capname* is not a numeric capability.

With the `tigetstr()` routine, the value `(char *)-1` is returned if *capname* is not a string capability.

The *capname* for each capability is given in the table column entitled *capname* code in the capabilities section of `terminfo(4)`.

```
char *boolnames, *boolcodes, *boolfnames
char *numnames, *numcodes, *numfnames
char *strnames, *strcodes, *strfnames
```

These null-terminated arrays contain the *capnames*, the *termcap* codes, and the full C names, for each of the *terminfo* variables.

RETURN VALUES

All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

Routines that return pointers always return `NULL` on error.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_initscr(3CURSES)`, `curs_kernel(3CURSES)`, `curs_termcap(3CURSES)`, `curses(3CURSES)`, `putc(3C)`, `terminfo(4)`, `attributes(5)`

mvcur(3CURSES)

NOTES | The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

The `setupterm()` routine should be used in place of `setterm()`.

Note that `vidattr()` and `vidputs()` may be macros.

NAME mvcur – move the cursor

SYNOPSIS `cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \`
`-R /usr/xpg4/lib -lcurses [library...]`

`c89 [flag...] file... -lcurses [library...]`

```
#include <curses.h>

int mvcur(int oldrow, int oldcol, int newrow, int newcol);
```

PARAMETERS

| | |
|---------------|---|
| <i>oldrow</i> | Is the row from which cursor is to be moved. |
| <i>oldcol</i> | Is the column from which cursor is to be moved. |
| <i>newrow</i> | Is the row to which cursor is to be moved. |
| <i>newcol</i> | Is the column to which cursor is to be moved. |

DESCRIPTION The mvcur() function is a low-level function used only outside of X/Open Curses when the program has to deal directly with the terminfo database to handle certain terminal capabilities. The use of appropriate X/Open Curses functions is recommended in all other situations, so that X/Open Curses can track the cursor.

The mvcur() function moves the cursor from the location specified by *oldrow* and *oldcol* to the location specified by *newrow* and *newcol*. A program using this function must keep track of the current cursor position.

RETURN VALUES On success, the mvcur() function returns OK. Otherwise, it returns ERR.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO libcurses(3XCURSES), attributes(5), standards(5)

mvdelch(3CURSES)

NAME curs_delch, delch, wdelch, mvdelch, mvwdelch – delete character under cursor in a curses window

SYNOPSIS

```
cc [ flag ... ] file ... -lcurses [ library ... ]
#include <curses.h>

int delch(void);
int wdelch(WINDOW *win);
int mvdelch(int y, int x);
int mvwdelch(WINDOW *win, int y, int x);
```

DESCRIPTION With these routines the character under the cursor in the window is deleted; all characters to the right of the cursor on the same line are moved to the left one position and the last character on the line is filled with a blank. The cursor position does not change (after moving to *y, x*, if specified). This does not imply use of the hardware delete character feature.

RETURN VALUES All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), attributes(5)

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `delch()`, `mvdelch()`, and `mvwdelch()` may be macros.

NAME delch, mvdelch, mvwdelch, wdelch – remove a character

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

int delch(void);

int mvdelch(int y, int x);

int mvwdelch(WINDOW *win, int y, int x);

int wdelch(WINDOW *win);
```

DESCRIPTION The delch() and wdelch() functions delete the character at the current cursor position from stdscr and win, respectively. All remaining characters after cursor through to the end of the line are shifted one character towards the start of the line. The last character on the line becomes a space; characters on other lines are not affected.

The mvdelch() and mvwdelch() functions delete the character at the position specified by the x and y parameters; the former deletes the character from stdscr; the latter from win.

PARAMETERS y Is the y (row) coordinate of the position of the character to be removed.
x Is the x (column) coordinate of the position of the character to be removed.
win Is a pointer to the window containing the character to be removed.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO bkgdset(3XCURSES), insch(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

mvderwin(3CURSES)

| | |
|--------------------|--|
| NAME | <code>curs_window</code> , <code>newwin</code> , <code>delwin</code> , <code>mvwin</code> , <code>subwin</code> , <code>derwin</code> , <code>mvderwin</code> , <code>dupwin</code> , <code>wsyncup</code> , <code>syncok</code> , <code>wcursyncup</code> , <code>wsyncdown</code> – create curses windows |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> WINDOW *newwin(int nlines, int ncols, int begin_y, int begin_x); int delwin(WINDOW *win); int mvwin(WINDOW *win, int y, int x); WINDOW *subwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); WINDOW *derwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); int mvderwin(WINDOW *win, int par_y, int par_x); WINDOW *dupwin(WINDOW *win); void wsyncup(WINDOW *win); int syncok(WINDOW *win, bool bf); void wcursyncup(WINDOW *win); void wsyncdown(WINDOW *win);</pre> |
| DESCRIPTION | <p>The <code>newwin()</code> routine creates and returns a pointer to a new window with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. The upper left-hand corner of the window is at line <i>begin_y</i>, column <i>begin_x</i>. If either <i>nlines</i> or <i>ncols</i> is zero, they default to <code>LINES</code> — <i>begin_y</i> and <code>COLS</code> — <i>begin_x</i>. A new full-screen window is created by calling <code>newwin(0, 0, 0, 0)</code>.</p> <p>The <code>delwin()</code> routine deletes the named window, freeing all memory associated with it. Subwindows must be deleted before the main window can be deleted.</p> <p>The <code>mvwin()</code> routine moves the window so that the upper left-hand corner is at position (<i>x</i>, <i>y</i>). If the move would cause the window to be off the screen, it is an error and the window is not moved. Moving subwindows is allowed, but should be avoided.</p> <p>The <code>subwin()</code> routine creates and returns a pointer to a new window with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. The window is at position (<i>begin_y</i>, <i>begin_x</i>) on the screen. (This position is relative to the screen, and not to the window <i>orig</i>.) The window is made in the middle of the window <i>orig</i>, so that changes made to one window will affect both windows. The subwindow shares memory with the window <i>orig</i>. When using this routine, it is necessary to call <code>touchwin()</code> or <code>touchline()</code> on <i>orig</i> before calling <code>wrefresh()</code> on the subwindow.</p> <p>The <code>derwin()</code> routine is the same as <code>subwin()</code>, except that <i>begin_y</i> and <i>begin_x</i> are relative to the origin of the window <i>orig</i> rather than the screen. There is no difference between the subwindows and the derived windows.</p> |

The `mvderwin()` routine moves a derived window (or subwindow) inside its parent window. The screen-relative parameters of the window are not changed. This routine is used to display different parts of the parent window at the same physical position on the screen.

The `dupwin()` routine creates an exact duplicate of the window *win*.

Each `curses` window maintains two data structures: the character image structure and the status structure. The character image structure is shared among all windows in the window hierarchy (that is, the window with all subwindows). The status structure, which contains information about individual line changes in the window, is private to each window. The routine `wrefresh()` uses the status data structure when performing screen updating. Since status structures are not shared, changes made to one window in the hierarchy may not be properly reflected on the screen.

The routine `wsyncup()` causes the changes in the status structure of a window to be reflected in the status structures of its ancestors. If `syncok()` is called with second argument `TRUE` then `wsyncup()` is called automatically whenever there is a change in the window.

The routine `wcursyncup()` updates the current cursor position of all the ancestors of the window to reflect the current cursor position of the window.

The routine `wsyncdown()` updates the status structure of the window to reflect the changes in the status structures of its ancestors. Applications seldom call this routine because it is called automatically by `wrefresh()`.

RETURN VALUES

Routines that return an integer return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`delwin()` returns the integer `ERR` upon failure and `OK` upon successful completion.

Routines that return pointers return `NULL` on error.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_refresh(3CURSES)`, `curs_touch(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

If many small changes are made to the window, the `wsyncup()` option could degrade performance.

`mvderwin(3CURSES)`

Note that `syncok()` may be a macro.

| NAME | mvderwin – map area of parent window to subwindow | | | | | | |
|----------------------|--|----------------|--|---------------------|--|--------------|---|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int mvderwin(WINDOW *win, int par_y, int par_x);</pre> | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>win</i></td> <td>Is a pointer to the window to be mapped.</td> </tr> <tr> <td><i>par_y</i></td> <td>Is the y (row) coordinate of the placement of the upper left corner of window relative to the parent window.</td> </tr> <tr> <td><i>par_x</i></td> <td>Is the x (column) coordinate of the placement of the upper left corner of the window relative to the parent window.</td> </tr> </table> | <i>win</i> | Is a pointer to the window to be mapped. | <i>par_y</i> | Is the y (row) coordinate of the placement of the upper left corner of window relative to the parent window. | <i>par_x</i> | Is the x (column) coordinate of the placement of the upper left corner of the window relative to the parent window. |
| <i>win</i> | Is a pointer to the window to be mapped. | | | | | | |
| <i>par_y</i> | Is the y (row) coordinate of the placement of the upper left corner of window relative to the parent window. | | | | | | |
| <i>par_x</i> | Is the x (column) coordinate of the placement of the upper left corner of the window relative to the parent window. | | | | | | |
| DESCRIPTION | <p>The <code>mvderwin()</code> function defines a mapped area of <i>win</i>'s parent window that is the same size as <i>win</i> and has its upper left corner at position <i>par_y</i>, <i>par_x</i> of the parent window.</p> <p>Whenever <i>win</i> is refreshed, its contents are updated to match those of the mapped area and any reference to characters in <i>win</i> is treated as a reference to corresponding characters in the mapped area.</p> | | | | | | |
| RETURN VALUES | On success, the <code>mvderwin()</code> function returns OK. Otherwise, it returns ERR. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>delwin(3XCURSES)</code> , <code>derwin(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

mvgetch(3CURSES)

| NAME | <code>curs_getch</code> , <code>getch</code> , <code>wgetch</code> , <code>mvgetch</code> , <code>mvwgetch</code> , <code>ungetch</code> – get (or push back) characters from curses terminal keyboard | | | | |
|------------------------|--|------|----------|------------------------|-----------|
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int getch(void); int wgetch(WINDOW *<i>win</i>); int mvgetch(int <i>y</i>, int <i>x</i>); int mvwgetch(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>); int ungetch(int <i>ch</i>);</pre> | | | | |
| DESCRIPTION | <p>With the <code>getch()</code>, <code>wgetch()</code>, <code>mvgetch()</code>, and <code>mvwgetch()</code> routines a character is read from the terminal associated with the window. In no-delay mode, if no input is waiting, the value <code>ERR</code> is returned. In delay mode, the program waits until the system passes text through to the program. Depending on the setting of <code>cbreak()</code>, this is after one character (<code>cbreak</code> mode), or after the first newline (<code>nocbreak</code> mode). In half-delay mode, the program waits until a character is typed or the specified timeout has been reached. Unless <code>noecho()</code> has been set, the character will also be echoed into the designated window.</p> <p>If the window is not a pad, and it has been moved or modified since the last call to <code>wrefresh()</code>, <code>wrefresh()</code> will be called before another character is read.</p> <p>If <code>keypad()</code> is <code>TRUE</code>, and a function key is pressed, the token for that function key is returned instead of the raw characters. Possible function keys are defined in <code><curses.h></code> with integers beginning with <code>0401</code>, whose names begin with <code>KEY_</code>. If a character that could be the beginning of a function key (such as escape) is received, <code>curses</code> sets a timer. If the remainder of the sequence does not come in within the designated time, the character is passed through; otherwise, the function key value is returned. For this reason, many terminals experience a delay between the time a user presses the escape key and the escape is returned to the program. Since tokens returned by these routines are outside the ASCII range, they are not printable.</p> <p>The <code>ungetch()</code> routine places <code>ch</code> back onto the input queue to be returned by the next call to <code>wgetch()</code>.</p> | | | | |
| Function Keys | <p>The following function keys, defined in <code><curses.h></code>, might be returned by <code>getch()</code> if <code>keypad()</code> has been enabled. Note that not all of these may be supported on a particular terminal if the terminal does not transmit a unique code when the key is pressed or if the definition for the key is not present in the <i>terminfo</i> database.</p> <table><thead><tr><th>Name</th><th>Key name</th></tr></thead><tbody><tr><td><code>KEY_BREAK</code></td><td>Break key</td></tr></tbody></table> | Name | Key name | <code>KEY_BREAK</code> | Break key |
| Name | Key name | | | | |
| <code>KEY_BREAK</code> | Break key | | | | |

| <i>Name</i> | <i>Key name</i> |
|-------------------|---|
| KEY_DOWN | The four arrow keys . . . |
| KEY_UP | |
| KEY_LEFT | |
| KEY_RIGHT | |
| KEY_HOME | Home key (upward+left arrow) |
| KEY_BACKSPACE | Backspace |
| KEY_F0 | Function keys; space for 64 keys is reserved. |
| KEY_F(<i>n</i>) | For $0 \leq n \leq 63$ |
| KEY_DL | Delete line |
| KEY_IL | Insert line |
| KEY_DC | Delete character |
| KEY_IC | Insert char or enter insert mode |
| KEY_EIC | Exit insert char mode |
| KEY_CLEAR | Clear screen |
| KEY_EOS | Clear to end of screen |
| KEY_EOL | Clear to end of line |
| KEY_SF | Scroll 1 line forward |
| KEY_SR | Scroll 1 line backward (reverse) |
| KEY_NPAGE | Next page |
| KEY_PPAGE | Previous page |
| KEY_STAB | Set tab |
| KEY_CTAB | Clear tab |
| KEY_CATAB | Clear all tabs |
| KEY_ENTER | Enter or send |
| KEY_SRESET | Soft (partial) reset |
| KEY_RESET | Reset or hard reset |
| KEY_PRINT | Print or copy |
| KEY_LL | Home down or bottom (lower left). Keypad is arranged like this: (Row 1) A1 up A3 (Row 2) left B2 right (Row 3) C1 down C3 |

mvgetch(3CURSES)

| <i>Name</i> | <i>Key name</i> |
|---------------|-----------------------|
| KEY_A1 | Upper left of keypad |
| KEY_A3 | Upper right of keypad |
| KEY_B2 | Center of keypad |
| KEY_C1 | Lower left of keypad |
| KEY_C3 | Lower right of keypad |
| KEY_BTAB | Back tab key |
| KEY_BEG | Beg(inning) key |
| KEY_CANCEL | Cancel key |
| KEY_CLOSE | Close key |
| KEY_COMMAND | Cmd (command) key |
| KEY_COPY | Copy key |
| KEY_CREATE | Create key |
| KEY_END | End key |
| KEY_EXIT | Exit key |
| KEY_FIND | Find key |
| KEY_HELP | Help key |
| KEY_MARK | Mark key |
| KEY_MESSAGE | Message key |
| KEY_MOVE | Move key |
| KEY_NEXT | Next object key |
| KEY_OPEN | Open key |
| KEY_OPTIONS | Options key |
| KEY_PREVIOUS | Previous object key |
| KEY_REDO | Redo key |
| KEY_REFERENCE | Reference key |
| KEY_REFRESH | Refresh key |
| KEY_REPLACE | Replace key |
| KEY_RESTART | Restart key |
| KEY_RESUME | Resume key |

| <i>Name</i> | <i>Key name</i> |
|---------------|-------------------------|
| KEY_SAVE | Save key |
| KEY_SBEG | Shifted beginning key |
| KEY_SCANCEL | Shifted cancel key |
| KEY_SCOMMAND | Shifted command key |
| KEY_SCOPY | Shifted copy key |
| KEY_SCREATE | Shifted create key |
| KEY_SDC | Shifted delete char key |
| KEY_SDL | Shifted delete line key |
| KEY_SELECT | Select key |
| KEY_SEND | Shifted end key |
| KEY_SEOL | Shifted clear line key |
| KEY_SEXIT | Shifted exit key |
| KEY_SFIND | Shifted find key |
| KEY_SHELP | Shifted help key |
| KEY_SHOME | Shifted home key |
| KEY_SIC | Shifted input key |
| KEY_SLEFT | Shifted left arrow key |
| KEY_SMESSAGE | Shifted message key |
| KEY_SMOVE | Shifted move key |
| KEY_SNEXT | Shifted next key |
| KEY_SOPTIONS | Shifted options key |
| KEY_SPREVIOUS | Shifted prev key |
| KEY_SPRINT | Shifted print key |
| KEY_SREDO | Shifted redo key |
| KEY_SREPLACE | Shifted replace key |
| KEY_SRIGHT | Shifted right arrow |
| KEY_SRSUME | Shifted resume key |
| KEY_SSAVE | Shifted save key |
| KEY_SSUSPEND | Shifted suspend key |

mvgetch(3CURSES)

| <i>Name</i> | <i>Key name</i> |
|-------------|------------------|
| KEY_SUNDO | Shifted undo key |
| KEY_SUSPEND | Suspend key |
| KEY_UNDO | Undo key |

RETURN VALUES All routines return the integer ERR upon failure. The `ungetch()` routine returns an integer value other than ERR upon successful completion. The other routines return the next input character or function key code upon successful completion.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_inopts(3CURSES)`, `curl_move(3CURSES)`, `curl_refresh(3CURSES)`, `curls(3CURSES)`, `attributes(5)`

NOTES The header `< curses.h >` automatically includes the headers `< stdio.h >` and `< unctrl.h >`.

Use of the escape key for a single character function is discouraged.

When using `getch()`, `wgetch()`, `mvgetch()`, or `mvwgetch()`, `nocbreak` mode (`nocbreak()`) and `echo` mode (`echo()`) should not be used at the same time. Depending on the state of the tty driver when each character is typed, the program may produce undesirable results.

Note that `getch()`, `mvgetch()`, and `mvwgetch()` may be macros.

| | |
|--|---|
| NAME | getch, wgetch, mvgetch, mvwgetch – get a single-byte character from the terminal |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int getch(void); int wgetch(WINDOW *win); int mvgetch(int y, int x); int mvwgetch(WINDOW *win, int y, int x);</pre> |
| PARAMETERS | <p><i>win</i> Is a pointer to the window associated with the terminal from which the character is to be read.</p> <p><i>y</i> Is the y (row) coordinate for the position of the character to be read.</p> <p><i>x</i> Is the x (column) coordinate for the position of the character to be read.</p> |
| DESCRIPTION | <p>These functions read a single-byte character from the terminal associated with the current or specified window. The results are unspecified if the input is not a single-byte character. If keypad(3XCURSES) is enabled, these functions respond to the pressing of a function key by returning the corresponding KEY_ value defined in <curses.h></p> <p>Processing of terminal input is subject to the general rules described on the keypad(3XCURSES) manual page.</p> <p>If echoing is enabled, then the character is echoed as though it were provided as an input argument to addch(3XCURSES), except for the following characters:</p> <p><backspace> The input is interpreted as follows: unless the cursor already was in column 0, <backspace> moves the cursor one column toward the start of the current line and any characters after the <backspace> are added or inserted starting there. The character at the resulting cursor position is then deleted as though delch(3XCURSES) were called, except that if the cursor was originally in the first column of the line, the user is alerted as though beep(3XCURSES) were called.</p> <p>Function keys The user is alerted as though beep() were called. Information concerning the function keys is not returned to the caller.</p> <p>If the current or specified window is not a pad, and it has been moved modified since the last refresh operation, then it will be refreshed before another character is read.</p> <p>The following is a list of tokens for function keys that are returned by the getch() set of functions if keypad handling is enabled (some terminals may not support all tokens).</p> |
| Constant Values for Function Keys | |

mvgetch(3XCURSES)

| Constant | Description |
|-------------------|---|
| KEY_BREAK | Break key |
| KEY_DOWN | The down arrow key |
| KEY_UP | The up arrow key |
| KEY_LEFT | The left arrow key |
| KEY_RIGHT | The right arrow key |
| KEY_HOME | Home key |
| KEY_BACKSPACE | Backspace |
| KEY_F0 | Function keys. Space for 64 keys is reserved. |
| KEY_F(<i>n</i>) | For $0 \leq n \leq 63$ |
| KEY_DL | Delete line |
| KEY_IL | Insert line |
| KEY_DC | Delete character |
| KEY_IC | Insert char or enter insert mode |
| KEY_EIC | Exit insert char mode |
| KEY_CLEAR | Clear screen |
| KEY_EOS | Clear to end of screen |
| KEY_EOL | Clear to end of line |
| KEY_SF | Scroll 1 line forward |
| KEY_SR | Scroll 1 line backwards |
| KEY_NPAGE | Next page |
| KEY_PPAGE | Previous page |
| KEY_STAB | Set tab |
| KEY_CTAB | Clear tab |
| KEY_CATAB | Clear all tabs |
| KEY_ENTER | Enter or send |
| KEY_SRESET | Soft (partial) reset |
| KEY_RESET | Reset or hard reset |
| KEY_PRINT | Print or copy |
| KEY_LL | Home down or bottom (lower left) |

| Constant | Description |
|---------------|-----------------------|
| KEY_A1 | Upper left of keypad |
| KEY_A3 | Upper right of keypad |
| KEY_B2 | Center of keypad |
| KEY_C1 | Lower left of keypad |
| KEY_C3 | Lower right of keypad |
| KEY_BTAB | Back tab |
| KEY_BEG | Beginning key |
| KEY_CANCEL | Cancel key |
| KEY_CLOSE | Close key |
| KEY_COMMAND | Cmd (command) key |
| KEY_COPY | Copy key |
| KEY_CREATE | Create key |
| KEY_END | End key |
| KEY_EXIT | Exit key |
| KEY_FIND | Find key |
| KEY_HELP | Help key |
| KEY_MARK | Mark key |
| KEY_MESSAGE | Message key |
| KEY_MOVE | Move key |
| KEY_NEXT | Next object key |
| KEY_OPEN | Open key |
| KEY_OPTIONS | Options key |
| KEY_PREVIOUS | Previous object key |
| KEY_REDO | Redo key |
| KEY_REFERENCE | Reference key |
| KEY_REFRESH | Refresh key |
| KEY_REPLACE | Replace key |
| KEY_RESTART | Restart key |
| KEY_RESUME | Resume key |

mvgetch(3XCURSES)

| Constant | Description |
|---------------|-------------------------|
| KEY_SAVE | Save key |
| KEY_SBEG | Shifted beginning key |
| KEY_SCANCEL | Shifted cancel key |
| KEY_SCOMMAND | Shifted command key |
| KEY_SCOPY | Shifted copy key |
| KEY_SCREATE | Shifted create key |
| KEY_SDC | Shifted delete char key |
| KEY_SDL | Shifted delete line key |
| KEY_SELECT | Select key |
| KEY_SEND | Shifted end key |
| KEY_SEOL | Shifted clear line key |
| KEY_SEXIT | Shifted exit key |
| KEY_SFIND | Shifted find key |
| KEY_SHELP | Shifted help key |
| KEY_SHOME | Shifted home key |
| KEY_SIC | Shifted input key |
| KEY_SLEFT | Shifted left arrow key |
| KEY_SMESSAGES | Shifted messages key |
| KEY_SMOVE | Shifted move key |
| KEY_SNEXT | Shifted next key |
| KEY_SOPTIONS | Shifted options key |
| KEY_SPREVIOUS | Shifted previous key |
| KEY_SPRINT | Shifted print key |
| KEY_SREDO | Shifted redo key |
| KEY_SREPLACE | Shifted replace key |
| KEY_SRIGHT | Shifted right arrow key |
| KEY_SRSUME | Shifted resume key |
| KEY_SSAVE | Shifted save key |
| KEY_SSUSPEND | Shifted suspend key |

| Constant | Description |
|-------------|------------------|
| KEY_SUNDO | Shifted undo key |
| KEY_SUSPEND | Suspend key |
| KEY_UNDO | Undo key |

RETURN VALUES Upon successful completion, these functions return the single-byte character, `KEY_` value, or `ERR`. When in the nodelay mode and no data is available, `ERR` is returned.

ERRORS No errors are defined.

USAGE Applications should not define the escape key by itself as a single-character function.

When using these functions, nocbreak mode (`cbreak(3XCURSES)`) and echo mode (`echo(3XCURSES)`) should not be used at the same time. Depending on the state of the terminal when each character is typed, the application may produce undesirable results.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `cbreak(3XCURSES)`, `echo(3XCURSES)`, `halfdelay(3XCURSES)`, `keypad(3XCURSES)`, `libcurses(3XCURSES)`, `nodelay(3XCURSES)`, `notimeout(3XCURSES)`, `raw(3XCURSES)`, `timeout(3XCURSES)`, `attributes(5)`, `standards(5)`

mvgetnstr(3XCURSES)

| | | | | | | | | | | | |
|--------------------|--|------------|--|----------|---|----------|--|----------|---|------------|---|
| NAME | getnstr, getstr, mvgetnstr, mvgetstr, mvwgetnstr, mvwgetstr, wgetnstr, wgetstr – get a multibyte character string from terminal | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int getnstr(char *str, int n); int getstr(char *str); int mvgetnstr(int y, int x, char *str, int n); int mvgetstr(int y, int x, char *str); int mvwgetnstr(WINDOW *win, int y, int x, char *str, int n); int mvwgetstr(WINDOW *win, int y, int x, char *str); int wgetnstr(WINDOW *win, char *str, int n); int wgetstr(WINDOW *win, char *str);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The <code>getstr()</code> and <code>wgetstr()</code> functions get a character string from the terminal associated with the window <code>stdscr</code> or window <code>win</code>, respectively. The <code>mvgetstr()</code> and <code>mvwgetstr()</code> functions move the cursor to the position specified in <code>stdscr</code> or <code>win</code>, respectively, then get a character string.</p> <p>These functions call <code>wgetch(3XCURSES)</code> and place each received character in <code>str</code> until a newline is received, which is also placed in <code>str</code>. The erase and kill characters set by the user are processed.</p> <p>The <code>getnstr()</code>, <code>mvgetnstr()</code>, <code>mvwgetnstr()</code> and <code>wgetnstr()</code> functions read at most <code>n</code> characters. These functions are used to prevent overflowing the input buffer.</p> <p>The <code>getnstr()</code>, <code>wgetnstr()</code>, <code>mvgetnstr()</code>, and <code>mvwgetnstr()</code> functions only return complete multibyte characters. If the area pointed to by <code>str</code> is not large enough to hold at least one character, these functions fail.</p> | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>str</i></td><td>Is a pointer to the area where the character string is to be placed.</td></tr><tr><td><i>n</i></td><td>Is the maximum number of characters to read from input.</td></tr><tr><td><i>y</i></td><td>Is the y (row) coordinate of starting position of character string to be read.</td></tr><tr><td><i>x</i></td><td>Is the x (column) coordinate of starting position of character string to be read.</td></tr><tr><td><i>win</i></td><td>Points to the window associated with the terminal from which the character is to be read.</td></tr></table> | <i>str</i> | Is a pointer to the area where the character string is to be placed. | <i>n</i> | Is the maximum number of characters to read from input. | <i>y</i> | Is the y (row) coordinate of starting position of character string to be read. | <i>x</i> | Is the x (column) coordinate of starting position of character string to be read. | <i>win</i> | Points to the window associated with the terminal from which the character is to be read. |
| <i>str</i> | Is a pointer to the area where the character string is to be placed. | | | | | | | | | | |
| <i>n</i> | Is the maximum number of characters to read from input. | | | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of starting position of character string to be read. | | | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of starting position of character string to be read. | | | | | | | | | | |
| <i>win</i> | Points to the window associated with the terminal from which the character is to be read. | | | | | | | | | | |

mvgetnstr(3XCURSES)

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO getch(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

mvgetnwstr(3CURSES)

| NAME | <code>curs_getwstr</code> , <code>getwstr</code> , <code>getnwstr</code> , <code>wgetwstr</code> , <code>wgetnwstr</code> , <code>mvgetwstr</code> , <code>mvgetnwstr</code> , <code>mvwgetwstr</code> , <code>mvwgetnwstr</code> – get <code>wchar_t</code> character strings from curses terminal keyboard | | | | |
|---------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int getwstr(wchar_t *wstr); int getnwstr(wchar_t *wstr, int n); int wgetwstr(WINDOW *win, wchar_t *wstr); int wgetnwstr(WINDOW *win, wchar_t *wstr, int n); int mvgetwstr(int y, int x, wchar_t *wstr); int mvgetnwstr(int y, int x, wchar_t *wstr, int n); int mvwgetwstr(WINDOW *win, int y, int x, wchar_t *wstr); int mvwgetnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);</pre> | | | | |
| DESCRIPTION | The effect of <code>getwstr()</code> is as though a series of calls to <code>getwch(3CURSES)</code> were made, until a newline and carriage return is received. The resulting value is placed in the area pointed to by the <code>wchar_t</code> pointer <code>wstr</code> . <code>getnwstr()</code> reads at most <code>n</code> <code>wchar_t</code> characters, thus preventing a possible overflow of the input buffer. The user's erase and kill characters are interpreted, as well as any special keys (such as function keys, HOME key, CLEAR key, etc.). | | | | |
| RETURN VALUE | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for a description of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>getwch(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header file <code><curses.h></code> automatically includes the header files <code><stdio.h></code> , <code><unctrl.h></code> , and <code><widec.h></code> . Note that all routines except <code>wgetnwstr()</code> may be macros. | | | | |

| | |
|----------------------|---|
| NAME | getn_wstr, get_wstr, mvgetn_wstr, mvget_wstr, mvwgetn_wstr, mvwget_wstr, wgetn_wstr, wget_wstr – get a wide character string from terminal |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int getn_wstr(wint_t *wstr, int n); int get_wstr(wint_t *wstr); int mvgetn_wstr(int y, int x, wint_t *wstr, int n); int mvget_wstr(int y, int x, wint_t *wstr); int mvwgetn_wstr(WINDOW *win, int y, int x, wint_t *wstr, int n); int mvwget_wstr(WINDOW *win, int y, int x, wint_t *wstr); int wgetn_wstr(WINDOW *win, wint_t *wstr, int n); int wget_wstr(WINDOW *win, wint_t *wstr);</pre> |
| DESCRIPTION | <p>The <code>get_wstr()</code> and <code>wget_wstr()</code> functions get a wide character string from the terminal associated with the window <code>stdscr</code> or window <code>win</code>, respectively. The <code>mvget_wstr()</code> and <code>mvwget_wstr()</code> functions move the cursor to the position specified in <code>stdscr</code> or <code>win</code>, respectively, then get a wide character string.</p> <p>These functions call <code>wget_wch(3XCURSES)</code> and place each received character in <code>wstr</code> until a newline character, end-of-line character, or end-of-file character is received, which is also placed in <code>wstr</code>. The erase and kill characters set by the user are processed.</p> <p>The <code>getn_wstr()</code>, <code>mvgetn_wstr()</code>, <code>mvwgetn_wstr()</code> and <code>wgetn_wstr()</code> functions read at most <code>n</code> characters. These functions are used to prevent overflowing the input buffer.</p> |
| PARAMETERS | <p><i>wstr</i> Is a pointer to the area where the character string is to be placed.</p> <p><i>n</i> Is the maximum number of characters to read from input.</p> <p><i>y</i> Is the y (row) coordinate of starting position of character string to be read.</p> <p><i>x</i> Is the x (column) coordinate of starting position of character string to be read.</p> <p><i>win</i> points to the window associated with the terminal from which the character is to be read.</p> |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. |
| ERRORS | None. |

mvgetn_wstr(3XCURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `get_wch(3XCURSES)`, `getnstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_getstr, getstr, wgetstr, mvgetstr, mvwgetstr, wgetnstr – get character strings from curses terminal keyboard | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int getstr(char *str); int wgetstr(WINDOW *win, char *str); int mvgetstr(int y, int x, char *str); int mvwgetstr(WINDOW *win, int y, int x, char *str); int wgetnstr(WINDOW *win, char *str, int n);</pre> | | | | |
| DESCRIPTION | The effect of <code>getstr()</code> is as though a series of calls to <code>getch()</code> were made, until a newline or carriage return is received. The resulting value is placed in the area pointed to by the character pointer <code>str</code> . <code>wgetnstr()</code> reads at most <code>n</code> characters, thus preventing a possible overflow of the input buffer. The user's erase and kill characters are interpreted, as well as any special keys (such as function keys, HOME key, and CLEAR key.) | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_getch(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that <code>getstr()</code>, <code>mvgetstr()</code>, and <code>mvwgetstr()</code> may be macros.</p> | | | | |

mvgetstr(3XCURSES)

| | | | | | | | | | | | |
|--------------------|--|------------|--|----------|---|----------|--|----------|---|------------|---|
| NAME | getnstr, getstr, mvgetnstr, mvgetstr, mvwgetnstr, mvwgetstr, wgetnstr, wgetstr – get a multibyte character string from terminal | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int getnstr(char *str, int n); int getstr(char *str); int mvgetnstr(int y, int x, char *str, int n); int mvgetstr(int y, int x, char *str); int mvwgetnstr(WINDOW *win, int y, int x, char *str, int n); int mvwgetstr(WINDOW *win, int y, int x, char *str); int wgetnstr(WINDOW *win, char *str, int n); int wgetstr(WINDOW *win, char *str);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The <code>getstr()</code> and <code>wgetstr()</code> functions get a character string from the terminal associated with the window <code>stdscr</code> or window <code>win</code>, respectively. The <code>mvgetstr()</code> and <code>mvwgetstr()</code> functions move the cursor to the position specified in <code>stdscr</code> or <code>win</code>, respectively, then get a character string.</p> <p>These functions call <code>wgetch(3XCURSES)</code> and place each received character in <code>str</code> until a newline is received, which is also placed in <code>str</code>. The erase and kill characters set by the user are processed.</p> <p>The <code>getnstr()</code>, <code>mvgetnstr()</code>, <code>mvwgetnstr()</code> and <code>wgetnstr()</code> functions read at most <code>n</code> characters. These functions are used to prevent overflowing the input buffer.</p> <p>The <code>getnstr()</code>, <code>wgetnstr()</code>, <code>mvgetnstr()</code>, and <code>mvwgetnstr()</code> functions only return complete multibyte characters. If the area pointed to by <code>str</code> is not large enough to hold at least one character, these functions fail.</p> | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>str</i></td><td>Is a pointer to the area where the character string is to be placed.</td></tr><tr><td><i>n</i></td><td>Is the maximum number of characters to read from input.</td></tr><tr><td><i>y</i></td><td>Is the y (row) coordinate of starting position of character string to be read.</td></tr><tr><td><i>x</i></td><td>Is the x (column) coordinate of starting position of character string to be read.</td></tr><tr><td><i>win</i></td><td>Points to the window associated with the terminal from which the character is to be read.</td></tr></table> | <i>str</i> | Is a pointer to the area where the character string is to be placed. | <i>n</i> | Is the maximum number of characters to read from input. | <i>y</i> | Is the y (row) coordinate of starting position of character string to be read. | <i>x</i> | Is the x (column) coordinate of starting position of character string to be read. | <i>win</i> | Points to the window associated with the terminal from which the character is to be read. |
| <i>str</i> | Is a pointer to the area where the character string is to be placed. | | | | | | | | | | |
| <i>n</i> | Is the maximum number of characters to read from input. | | | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of starting position of character string to be read. | | | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of starting position of character string to be read. | | | | | | | | | | |
| <i>win</i> | Points to the window associated with the terminal from which the character is to be read. | | | | | | | | | | |

mvgetstr(3XCURSES)

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO getch(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

mvgetwch(3CURSES)

| NAME | <code> curs_getwch, getwch, wgetwch, mvgetwch, mvwgetwch, ungetwch </code> – get (or push back) <code>wchar_t</code> characters from curses terminal keyboard | | | | |
|------------------------|--|------|----------|------------------------|-----------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int getwch(void); int wgetwch(WINDOW *win); int mvgetwch(int y, int x); int mvwgetwch(WINDOW *win, int y, int x); int ungetwch(int wch);</pre> | | | | |
| DESCRIPTION | <p>The <code>getwch()</code>, <code>wgetwch()</code>, <code>mvgetwch()</code>, and <code>mvwgetwch()</code> routines read an EUC character from the terminal associated with the window, transform it into a <code>wchar_t</code> character, and return a <code>wchar_t</code> character. In no-delay mode, if no input is waiting, the value <code>ERR</code> is returned. In delay mode, the program waits until the system passes text through to the program. Depending on the setting of <code>cbreak</code>, this is after one character (<code>cbreak</code> mode), or after the first newline (<code>nocbreak</code> mode). In half-delay mode, the program waits until a character is typed or the specified timeout has been reached. Unless <code>noecho</code> has been set, the character will also be echoed into the designated window.</p> <p>If the window is not a pad, and it has been moved or modified since the last call to <code>wrefresh(3CURSES)</code>, <code>wrefresh</code> will be called before another character is read.</p> <p>If <code>keypad</code> is <code>TRUE</code>, and a function key is pressed, the token for that function key is returned instead of the raw characters. Possible function keys are defined in <code><curses.h></code> with integers beginning with <code>0401</code>, whose names begin with <code>KEY_</code>. If a character that could be the beginning of a function key (such as escape) is received, <code>curses(3CURSES)</code> sets a timer. If the remainder of the sequence does not come in within the designated time, the character is passed through; otherwise, the function key value is returned. For this reason, many terminals experience a delay between the time a user presses the escape key and the escape is returned to the program.</p> <p>The <code>ungetwch()</code> routine places <code>wch</code> back onto the input queue to be returned by the next call to <code>wgetwch()</code>.</p> | | | | |
| Function Keys | <p>The following function keys, defined in <code><curses.h></code>, might be returned by <code>getwch()</code> if <code>keypad</code> has been enabled. Note that not all of these may be supported on a particular terminal if the terminal does not transmit a unique code when the key is pressed or if the definition for the key is not present in the <code>terminfo(4)</code> database.</p> <table border="1"><thead><tr><th>Name</th><th>Key name</th></tr></thead><tbody><tr><td><code>KEY_BREAK</code></td><td>Break key</td></tr></tbody></table> | Name | Key name | <code>KEY_BREAK</code> | Break key |
| Name | Key name | | | | |
| <code>KEY_BREAK</code> | Break key | | | | |

| <i>Name</i> | <i>Key name</i> |
|-------------------|---|
| KEY_DOWN | The four arrow keys . . . |
| KEY_UP | |
| KEY_LEFT | |
| KEY_RIGHT | |
| KEY_HOME | Home key (upward+left arrow) |
| KEY_BACKSPACE | Backspace |
| KEY_F0 | Function keys; space for 64 keys is reserved. |
| KEY_F(<i>n</i>) | For $0 \leq n \leq 63$ |
| KEY_DL | Delete line |
| KEY_IL | Insert line |
| KEY_DC | Delete character |
| KEY_IC | Insert char or enter insert mode |
| KEY_EIC | Exit insert char mode |
| KEY_CLEAR | Clear screen |
| KEY_EOS | Clear to end of screen |
| KEY_EOL | Clear to end of line |
| KEY_SF | Scroll 1 line forward |
| KEY_SR | Scroll 1 line backward (reverse) |
| KEY_NPAGE | Next page |
| KEY_PPAGE | Previous page |
| KEY_STAB | Set tab |
| KEY_CTAB | Clear tab |
| KEY_CATAB | Clear all tabs |
| KEY_ENTER | Enter or send |
| KEY_SRESET | Soft (partial) reset |
| KEY_RESET | Reset or hard reset |
| KEY_PRINT | Print or copy |
| KEY_LL | Home down or bottom (lower left). Keypad is arranged like this: A1 up A3 left B2 right C1 down C3 |

mvgetwch(3CURSES)

| <i>Name</i> | <i>Key name</i> |
|---------------|-----------------------|
| KEY_A1 | Upper left of keypad |
| KEY_A3 | Upper right of keypad |
| KEY_B2 | Center of keypad |
| KEY_C1 | Lower left of keypad |
| KEY_C3 | Lower right of keypad |
| KEY_BTAB | Back tab key |
| KEY_BEG | Beg(inning) key |
| KEY_CANCEL | Cancel key |
| KEY_CLOSE | Close key |
| KEY_COMMAND | Cmd (command) key |
| KEY_COPY | Copy key |
| KEY_CREATE | Create key |
| KEY_END | End key |
| KEY_EXIT | Exit key |
| KEY_FIND | Find key |
| KEY_HELP | Help key |
| KEY_MARK | Mark key |
| KEY_MESSAGE | Message key |
| KEY_MOVE | Move key |
| KEY_NEXT | Next object key |
| KEY_OPEN | Open key |
| KEY_OPTIONS | Options key |
| KEY_PREVIOUS | Previous object key |
| KEY_REDO | Redo key |
| KEY_REFERENCE | Reference key |
| KEY_REFRESH | Refresh key |
| KEY_REPLACE | Replace key |
| KEY_RESTART | Restart key |
| KEY_RESUME | Resume key |

| <i>Name</i> | <i>Key name</i> |
|---------------|-------------------------|
| KEY_SAVE | Save key |
| KEY_SBEG | Shifted beginning key |
| KEY_SCANCEL | Shifted cancel key |
| KEY_SCOMMAND | Shifted command key |
| KEY_SCOPY | Shifted copy key |
| KEY_SCREATE | Shifted create key |
| KEY_SDC | Shifted delete char key |
| KEY_SDL | Shifted delete line key |
| KEY_SELECT | Select key |
| KEY_SEND | Shifted end key |
| KEY_SEOL | Shifted clear line key |
| KEY_SEXIT | Shifted exit key |
| KEY_SFIND | Shifted find key |
| KEY_SHELP | Shifted help key |
| KEY_SHOME | Shifted home key |
| KEY_SIC | Shifted input key |
| KEY_SLEFT | Shifted left arrow key |
| KEY_SMESSAGE | Shifted message key |
| KEY_SMOVE | Shifted move key |
| KEY_SNEXT | Shifted next key |
| KEY_SOPTIONS | Shifted options key |
| KEY_SPREVIOUS | Shifted prev key |
| KEY_SPRINT | Shifted print key |
| KEY_SREDO | Shifted redo key |
| KEY_SREPLACE | Shifted replace key |
| KEY_SRIGHT | Shifted right arrow |
| KEY_SRSUME | Shifted resume key |
| KEY_SSAVE | Shifted save key |
| KEY_SSUSPEND | Shifted suspend key |

mvgetwch(3CURSES)

| <i>Name</i> | <i>Key name</i> |
|-------------|------------------|
| KEY_SUNDO | Shifted undo key |
| KEY_SUSPEND | Suspend key |
| KEY_UNDO | Undo key |

RETURN VALUE All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3CURSES)`, `curl_inopts(3CURSES)`, `curl_move(3CURSES)`, `wrefresh(3CURSES)`, `terminfo(4)`, `attributes(5)`

NOTES The header file `<curses.h>` automatically includes the header files `<stdio.h>`, `<unctrl.h>` and `<widec.h>`.

Use of the escape key by a programmer for a single character function is discouraged.

When using `getwch()`, `wgetwch()`, `mvgetwch()`, or `mvwgetwch()`, `nocbreak` mode and `echo` mode should not be used at the same time. Depending on the state of the tty driver when each character is typed, the program may produce undesirable results.

Note that `getwch()`, `mvgetwch()`, and `mvwgetwch()` may be macros.

| | |
|--------------------|---|
| NAME | get_wch, wget_wch, mvget_wch, mvwget_wch – get a wide character from terminal |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int get_wch(wint_t *ch); int wget_wch(WINDOW *win, wint_t *ch); int mvget_wch(int y, int x, wint_t *ch); int mvwget_wch(WINDOW *win, int y, int x, wint_t *ch);</pre> |
| DESCRIPTION | <p>The <code>get_wch()</code> and <code>wget_wch()</code> functions get a wide character from the terminal associated with the window <code>stdscr</code> or window <code>win</code>, respectively. The <code>mvget_wch()</code> and <code>mvwget_wch()</code> functions move the cursor to the position specified in <code>stdscr</code> or <code>win</code>, respectively, then get a character.</p> <p>If the window is not a pad and has been changed since the last call to <code>refresh(3XCURSES)</code>, <code>get_wch()</code> calls <code>refresh()</code> to update the window before the next character is read.</p> <p>The setting of certain functions affects the behavior of the <code>get_wch()</code> set of functions. For example, if <code>cbreak(3XCURSES)</code> is set, characters typed by the user are immediately processed. If <code>halfdelay(3XCURSES)</code> is set, <code>get_wch()</code> waits until a character is typed or returns <code>ERR</code> if no character is typed within the specified timeout period. This timeout can also be specified for individual windows with the <code>delay</code> parameter of <code>timeout(3XCURSES)</code>. A negative value waits for input; a value of 0 returns <code>ERR</code> if no input is ready; a positive value blocks until input arrives or the time specified expires (in which case <code>ERR</code> is returned). If <code>nodelay(3XCURSES)</code> is set, <code>ERR</code> is returned if no input is waiting; if not set, <code>get_wch()</code> waits until input arrives. Each character will be echoed to the window unless <code>noecho(3XCURSES)</code> has been set.</p> <p>If keypad handling is enabled (<code>keypad(3XCURSES)</code> is <code>TRUE</code>), the token for the function key (a <code>KEY_</code> value) is stored in the object pointed to by <code>ch</code> and <code>KEY_CODE_YES</code> is returned. If a character is received that could be the beginning of a function key (for example, <code>ESC</code>), an inter-byte timer is set. If the remainder of the sequence is not received before the time expires, the character is passed through; otherwise, the value of the function key is returned. If <code>notimeout()</code> is set, the inter-byte timer is not used.</p> <p>The <code>ESC</code> key is typically a prefix key used with function keys and should not be used as a single character.</p> <p>See the <code>getch(3XCURSES)</code> manual page for a list of tokens for function keys that are returned by the <code>get_wch()</code> set of functions if keypad handling is enabled (Some terminals may not support all tokens).</p> |

`mvget_wch(3XCURSES)`

PARAMETERS

| | |
|------------|--|
| <i>ch</i> | Is a pointer to a wide integer where the returned wide character or <code>KEY_</code> value can be stored. |
| <i>win</i> | Is a pointer to the window associated with the terminal from which the character is to be read. |
| <i>y</i> | Is the y (row) coordinate for the position of the character to be read. |
| <i>x</i> | Is the x (column) coordinate for the position of the character to be read. |

RETURN VALUES When these functions successfully report the pressing of a function key, they return `KEY_CODE_YES`. When they successfully report a wide character, they return `OK`. Otherwise, they return `ERR`.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `cbreak(3XCURSES)`, `echo(3XCURSES)`, `halfdelay(3XCURSES)`, `keypad(3XCURSES)`, `libcurses(3XCURSES)`, `nodelay(3XCURSES)`, `notimeout(3XCURSES)`, `raw(3XCURSES)`, `timeout(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_getwstr, getwstr, getnwstr, wgetwstr, wgetnwstr, mvgetwstr, mvgetnwstr, mvwgetwstr, mvwgetnwstr – get wchar_t character strings from curses terminal keyboard | | | | |
|---------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int getwstr(wchar_t *wstr); int getnwstr(wchar_t *wstr, int n); int wgetwstr(WINDOW *win, wchar_t *wstr); int wgetnwstr(WINDOW *win, wchar_t *wstr, int n); int mvgetwstr(int y, int x, wchar_t *wstr); int mvgetnwstr(int y, int x, wchar_t *wstr, int n); int mvwgetwstr(WINDOW *win, int y, int x, wchar_t *wstr); int mvwgetnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);</pre> | | | | |
| DESCRIPTION | The effect of getwstr() is as though a series of calls to getch(3CURSES) were made, until a newline and carriage return is received. The resulting value is placed in the area pointed to by the wchar_t pointer wstr. getnwstr() reads at most n wchar_t characters, thus preventing a possible overflow of the input buffer. The user's erase and kill characters are interpreted, as well as any special keys (such as function keys, HOME key, CLEAR key, etc.). | | | | |
| RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See attributes(5) for a description of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), getch(3CURSES), attributes(5) | | | | |
| NOTES | <p>The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h>, and <wchar.h>.</p> <p>Note that all routines except wgetnwstr() may be macros.</p> | | | | |

mvget_wstr(3XCURSES)

| | | | | | | | | | | | |
|----------------------|---|-------------|--|----------|---|----------|--|----------|---|------------|---|
| NAME | getn_wstr, get_wstr, mvgetn_wstr, mvget_wstr, mvwgetn_wstr, mvwget_wstr, wgetn_wstr, wget_wstr – get a wide character string from terminal | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int getn_wstr(wint_t *wstr, int n); int get_wstr(wint_t *wstr); int mvgetn_wstr(int y, int x, wint_t *wstr, int n); int mvget_wstr(int y, int x, wint_t *wstr); int mvwgetn_wstr(WINDOW *win, int y, int x, wint_t *wstr, int n); int mvwget_wstr(WINDOW *win, int y, int x, wint_t *wstr); int wgetn_wstr(WINDOW *win, wint_t *wstr, int n); int wget_wstr(WINDOW *win, wint_t *wstr);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The <code>get_wstr()</code> and <code>wget_wstr()</code> functions get a wide character string from the terminal associated with the window <code>stdscr</code> or window <code>win</code>, respectively. The <code>mvget_wstr()</code> and <code>mvwget_wstr()</code> functions move the cursor to the position specified in <code>stdscr</code> or <code>win</code>, respectively, then get a wide character string.</p> <p>These functions call <code>wget_wch(3XCURSES)</code> and place each received character in <code>wstr</code> until a newline character, end-of-line character, or end-of-file character is received, which is also placed in <code>wstr</code>. The erase and kill characters set by the user are processed.</p> <p>The <code>getn_wstr()</code>, <code>mvgetn_wstr()</code>, <code>mvwgetn_wstr()</code> and <code>wgetn_wstr()</code> functions read at most <code>n</code> characters. These functions are used to prevent overflowing the input buffer.</p> | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>wstr</i></td><td>Is a pointer to the area where the character string is to be placed.</td></tr><tr><td><i>n</i></td><td>Is the maximum number of characters to read from input.</td></tr><tr><td><i>y</i></td><td>Is the y (row) coordinate of starting position of character string to be read.</td></tr><tr><td><i>x</i></td><td>Is the x (column) coordinate of starting position of character string to be read.</td></tr><tr><td><i>win</i></td><td>points to the window associated with the terminal from which the character is to be read.</td></tr></table> | <i>wstr</i> | Is a pointer to the area where the character string is to be placed. | <i>n</i> | Is the maximum number of characters to read from input. | <i>y</i> | Is the y (row) coordinate of starting position of character string to be read. | <i>x</i> | Is the x (column) coordinate of starting position of character string to be read. | <i>win</i> | points to the window associated with the terminal from which the character is to be read. |
| <i>wstr</i> | Is a pointer to the area where the character string is to be placed. | | | | | | | | | | |
| <i>n</i> | Is the maximum number of characters to read from input. | | | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of starting position of character string to be read. | | | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of starting position of character string to be read. | | | | | | | | | | |
| <i>win</i> | points to the window associated with the terminal from which the character is to be read. | | | | | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | | | | | |
| ERRORS | None. | | | | | | | | | | |

mvget_wstr(3XCURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `get_wch(3XCURSES)`, `getnstr(3XCURSES)`, `libcurses(3XCURSES)`,
`attributes(5)`, `standards(5)`

mvhline(3XCURSES)

| | |
|----------------------|--|
| NAME | <code>hline</code> , <code>mvhline</code> , <code>mvvline</code> , <code>mvwhline</code> , <code>mvwvline</code> , <code>vline</code> , <code>whline</code> , <code>wvline</code> – use single-byte characters (and renditions) to draw lines |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int hline(chtype ch, int n); int mvhline(int y, int x, chtype ch, int n); int mvvline(int y, int x, chtype ch, int n); int mvwhline(WINDOW *win, int y, int x, chtype ch, int n); int mvwvline(WINDOW *win, int y, int x, chtype ch, int n); int vline(chtype ch, int n); int whline(WINDOW *win, chtype ch, int n); int wvline(WINDOW *win, chtype ch, int n);</pre> |
| DESCRIPTION | <p>The <code>hline()</code>, <code>vline()</code>, <code>whline()</code>, <code>wvline()</code> functions draw a horizontal or vertical line, in either the window <code>stdscr</code> or <code>win</code> starting at the current cursor position. The line is drawn using the character <code>ch</code> and is a maximum of <code>n</code> positions long, or as many as will fit into the window. If <code>ch</code> is 0 (zero), the default horizontal or vertical character is used.</p> <p>The <code>mvhline()</code>, <code>mvvline()</code>, <code>mvwhline()</code>, <code>mvwvline()</code> functions are similar to the previous group of functions but the line begins at cursor position specified by <code>x</code> and <code>y</code>.</p> <p>The functions with names ending with <code>hline()</code> draw horizontal lines proceeding towards the last column of the same line. The functions with names ending with <code>vline()</code> draw vertical lines proceeding towards the last column of the same line.</p> <p>These functions do not change the position of the cursor.</p> |
| PARAMETERS | <p><code>ch</code> Is the character used to draw the line.</p> <p><code>n</code> Is the maximum number of characters in the line.</p> <p><code>y</code> Is the <code>y</code> (row) coordinate for the start of the line.</p> <p><code>x</code> Is the <code>x</code> (column) coordinate for the start of the line.</p> <p><code>win</code> Is a pointer to a window.</p> |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. |
| ERRORS | None |

`mvhline(3XCURSES)`

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `border(3XCURSES)`, `border_set(3XCURSES)`, `hline_set(3XCURSES)`,
`libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

mvhline_set(3XCURSES)

| | | | | | | | | | | | |
|----------------------|---|------------------|---|----------------|--|----------------|--|----------------|---|------------------|---------------------------|
| NAME | <code>hline_set</code> , <code>mvhline_set</code> , <code>mvvline_set</code> , <code>mvwhline_set</code> , <code>mvwvline_set</code> , <code>vline_set</code> , <code>whline_set</code> , <code>wvline_set</code> – use complex characters (and renditions) to draw lines | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int hline_set(const cchar_t *ch, int n); int mvhline_set(int y, int x, const cchar_t *wch, int n); int mvvline_set(int y, int x, const cchar_t *wch, int n); int mvwhline_set(WINDOW *win, int y, int x, const cchar_t *wch, int n); int mvwvline_set(WINDOW *win, int y, int x, const cchar_t *wch, int n); int vline_set(const cchar_t *wch, int n); int whline_set(WINDOW *win, const cchar_t *wch, int n); int wvline_set(WINDOW *win, const cchar_t *wch, int n);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The <code>hline_set()</code>, <code>vline_set()</code>, <code>whline_set()</code>, <code>wvline_set()</code> functions draw a line, in either the window <code>stdscr</code> or <code>win</code> starting at the current cursor position. The line is drawn using the character <code>wch</code> and is a maximum of <code>n</code> positions long, or as many as will fit into the window. If <code>wch</code> is a null pointer, the default horizontal or vertical character is used.</p> <p>The <code>mvhline_set()</code>, <code>mvvline_set()</code>, <code>mvwhline_set()</code>, <code>mvwvline_set()</code> functions are similar to the previous group of functions but the line begins at cursor position specified by <code>x</code> and <code>y</code>.</p> <p>The functions with names ending with <code>hline_set()</code> draw horizontal lines proceeding towards the last column of the same line. The functions with names ending with <code>vline_set()</code> draw vertical lines proceeding towards the last column of the same line.</p> <p>These functions do not change the position of the cursor.</p> | | | | | | | | | | |
| PARAMETERS | <table><tr><td><code>wch</code></td><td>Is the complex character used to draw the line.</td></tr><tr><td><code>n</code></td><td>Is the maximum number of characters in the line.</td></tr><tr><td><code>y</code></td><td>Is the y (row) coordinate for the start of the line.</td></tr><tr><td><code>x</code></td><td>Is the x (column) coordinate for the start of the line.</td></tr><tr><td><code>win</code></td><td>Is a pointer to a window.</td></tr></table> | <code>wch</code> | Is the complex character used to draw the line. | <code>n</code> | Is the maximum number of characters in the line. | <code>y</code> | Is the y (row) coordinate for the start of the line. | <code>x</code> | Is the x (column) coordinate for the start of the line. | <code>win</code> | Is a pointer to a window. |
| <code>wch</code> | Is the complex character used to draw the line. | | | | | | | | | | |
| <code>n</code> | Is the maximum number of characters in the line. | | | | | | | | | | |
| <code>y</code> | Is the y (row) coordinate for the start of the line. | | | | | | | | | | |
| <code>x</code> | Is the x (column) coordinate for the start of the line. | | | | | | | | | | |
| <code>win</code> | Is a pointer to a window. | | | | | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | | | | | |

ERRORS None.**ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `border(3XCURSES)`, `border_set(3XCURSES)`, `hline(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

mvinch(3CURSES)

- NAME** curs_inch, inch, winch, mvinch, mvwinch – get a character and its attributes from a curses window
- SYNOPSIS**

```
cc [ flag ... ] file ... -lcurses [ library ... ]
#include <curses.h>

ctype inch(void);
ctype winch(WINDOW *win);
ctype mvinch(int y, int x);
ctype mvwinch(WINDOW *win, int y, int x);
```
- DESCRIPTION** With these routines, the character, of type `ctype`, at the current position in the named window is returned. If any attributes are set for that position, their values are OR-ed into the value returned. Constants defined in `<curses.h>` can be used with the logical AND (&) operator to extract the character or attributes alone.
- Attributes** The following bit-masks can be AND-ed with characters returned by `winch()`.
- | | |
|--------------|--|
| A_CHARTEXT | Bit-mask to extract character |
| A_ATTRIBUTES | Bit-mask to extract attributes |
| A_COLOR | Bit-mask to extract color-pair field information |
- ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:
- | ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |
- SEE ALSO** `curses(3CURSES)`, `attributes(5)`
- NOTES** The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.
- Note that all of these routines may be macros.

NAME | inch, mvinch, mvwinch, winch – return a single-byte character (with rendition)

SYNOPSIS |

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

ctype inch(void);

ctype mvinch(int y, int x);

ctype mvwinch(WINDOW *win, int y, int x);

ctype winch(WINDOW *win);
```

DESCRIPTION | The `inch()` and `winch()` functions return the `ctype` character located at the current cursor position of the `stdscr` window and window `win`, respectively. The `mvinch()` and `mvwinch()` functions return the `ctype` character located at the position indicated by the `x` (column) and `y` (row) parameters (the former in the `stdscr` window; the latter in window `win`).

The complete character/attribute pair will be returned. The character or attributes can be extracted by performing a bitwise AND on the returned value, using the constants `A_CHARTEXT`, `A_ATTRIBUTES`, and `A_COLOR`.

PARAMETERS | `y` | Is the `y` (row) coordinate of the position of the character to be returned.
`x` | Is the `x` (column) coordinate of the position of the character to be returned.
`win` | Is a pointer to the window that contains the character to be returned.

RETURN VALUES | On success, these functions return the specified character and rendition. Otherwise, they return `ERR`.

ERRORS | None.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | `addch(3XCURSES)`, `attroff(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

`mvinchnstr(3CURSES)`

| NAME | <code> curs_ inchstr, inchstr, inchnstr, winchstr, winchnstr, mvinchstr, mvinchnstr, mvwinchstr, mvwinchnstr</code> – get a string of characters (and attributes) from a curses window | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int inchstr(chtype *chstr); int inchnstr(chtype *chstr, int n); int winchstr(WINDOW *win, chtype *chstr); int winchnstr(WINDOW *win, chtype *chstr, int n); int mvinchstr(int y, int x, chtype *chstr); int mvinchnstr(int y, int x, chtype *chstr, int n); int mvwinchstr(WINDOW *win, int y, int x, chtype *chstr); int mvwinchnstr(WINDOW *win, int y, int x, chtype *chstr, int n);</pre> | | | | |
| DESCRIPTION | With these routines, a string of type <code>chtype</code> , starting at the current cursor position in the named window and ending at the right margin of the window, is returned. The four functions with <code>n</code> as the last argument, return the string at most <code>n</code> characters long. Constants defined in <code><curses.h></code> can be used with the <code>&</code> (logical AND) operator to extract the character or the attribute alone from any position in the <code>chstr</code> (see <code>curs_ inch(3CURSES)</code>). | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">MT-Level</td> <td style="text-align: center;">Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_ inch(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that all routines except <code>winchnstr()</code> may be macros.</p> | | | | |

| | |
|--------------------|--|
| NAME | <p>inchstr, inchstr, mvinchnstr, mvinchstr, mvwinchnstr, mvwinchstr, winchnstr, winchstr – retrieve a single-byte character string (with rendition)</p> |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int inchstr(chtype *chstr, int n); int inchstr(chtype *chstr); int mvinchnstr(int y, int x, chtype *chstr, int n); int mvinchstr(int y, int x, chtype *chstr); int mvwinchnstr(WINDOW *win, int y, int x, chtype *chstr, int n); int mvwinchstr(WINDOW *win, int y, int x, chtype *chstr); int winchnstr(WINDOW *win, chtype *chstr, int n); int winchstr(WINDOW *win, chtype *chstr);</pre> |
| DESCRIPTION | <p>The <code>inchstr()</code> and <code>winchstr()</code> functions retrieve the character string (with rendition) starting at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively, and ending at the right margin. The <code>mvinchstr()</code> and <code>mvwinchstr()</code> functions retrieve the character string located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>inchstr()</code>, <code>winchnstr()</code>, <code>mvinchnstr()</code>, and <code>mvwinchnstr()</code> functions retrieve at most <i>n</i> characters from the window <code>stdscr</code> and <i>win</i>, respectively. The former two functions retrieve the string, starting at the current cursor position; the latter two commands retrieve the string, starting at the position specified by the <i>x</i> and <i>y</i> parameters.</p> <p>All these functions store the retrieved character string in the object pointed to by <i>chstr</i>.</p> <p>The complete character/attribute pair is retrieved. The character or attributes can be extracted by performing a bitwise AND on the retrieved value, using the constants <code>A_CHARTEXT</code>, <code>A_ATTRIBUTES</code>, and <code>A_COLOR</code>. The character string can also be retrieved without attributes by using <code>instr(3XCURSES)</code> set of functions.</p> |
| PARAMETERS | <p><i>chstr</i> Is a pointer to an object that can hold the retrieved character string.</p> <p><i>n</i> Is the number of characters not to exceed when retrieving <i>chstr</i>.</p> <p><i>y</i> Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved.</p> <p><i>x</i> Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved.</p> |

`mvinchnstr(3XCURSES)`

win Is a pointer to the window in which the string is to be retrieved.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `inch(3XCURSES)`, `innstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

NAME | curs_inchstr, inchstr, inchnstr, winchstr, winchnstr, mvinchstr, mvinchnstr, mvwinchstr, mvwinchnstr – get a string of characters (and attributes) from a curses window

SYNOPSIS | `cc [flag ...] file ... -lcurses [library ...]`
`#include <curses.h>`
`int inchstr(chtype *chstr);`
`int inchnstr(chtype *chstr, int n);`
`int winchstr(WINDOW *win, chtype *chstr);`
`int winchnstr(WINDOW *win, chtype *chstr, int n);`
`int mvinchstr(int y, int x, chtype *chstr);`
`int mvinchnstr(int y, int x, chtype *chstr, int n);`
`int mvwinchstr(WINDOW *win, int y, int x, chtype *chstr);`
`int mvwinchnstr(WINDOW *win, int y, int x, chtype *chstr, int n);`

DESCRIPTION | With these routines, a string of type chtype, starting at the current cursor position in the named window and ending at the right margin of the window, is returned. The four functions with *n* as the last argument, return the string at most *n* characters long. Constants defined in <curses.h> can be used with the & (logical AND) operator to extract the character or the attribute alone from any position in the *chstr* (see curs_inch(3CURSES)).

RETURN VALUES | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | curs_inch(3CURSES), curses(3CURSES), attributes(5)

NOTES | The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.

Note that all routines except winchnstr() may be macros.

mvinchstr(3XCURSES)

| | | | | | | | | | |
|--------------------|---|--------------|---|----------|--|----------|--|----------|---|
| NAME | <code>inchstr</code> , <code>inchstr</code> , <code>mvinchstr</code> , <code>mvinchstr</code> , <code>mvwinchstr</code> , <code>mvwinchstr</code> , <code>winchstr</code> , <code>winchstr</code> – retrieve a single-byte character string (with rendition) | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int inchstr(chtype *chstr, int n); int inchstr(chtype *chstr); int mvinchstr(int y, int x, chtype *chstr, int n); int mvinchstr(int y, int x, chtype *chstr); int mvwinchstr(WINDOW *win, int y, int x, chtype *chstr, int n); int mvwinchstr(WINDOW *win, int y, int x, chtype *chstr); int winchstr(WINDOW *win, chtype *chstr, int n); int winchstr(WINDOW *win, chtype *chstr);</pre> | | | | | | | | |
| DESCRIPTION | <p>The <code>inchstr()</code> and <code>winchstr()</code> functions retrieve the character string (with rendition) starting at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively, and ending at the right margin. The <code>mvinchstr()</code> and <code>mvwinchstr()</code> functions retrieve the character string located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>inchstr()</code>, <code>winchstr()</code>, <code>mvinchstr()</code>, and <code>mvwinchstr()</code> functions retrieve at most <i>n</i> characters from the window <code>stdscr</code> and <i>win</i>, respectively. The former two functions retrieve the string, starting at the current cursor position; the latter two commands retrieve the string, starting at the position specified by the <i>x</i> and <i>y</i> parameters.</p> <p>All these functions store the retrieved character string in the object pointed to by <i>chstr</i>.</p> <p>The complete character/attribute pair is retrieved. The character or attributes can be extracted by performing a bitwise AND on the retrieved value, using the constants <code>A_CHARTEXT</code>, <code>A_ATTRIBUTES</code>, and <code>A_COLOR</code>. The character string can also be retrieved without attributes by using <code>instr(3XCURSES)</code> set of functions.</p> | | | | | | | | |
| PARAMETERS | <table><tr><td><i>chstr</i></td><td>Is a pointer to an object that can hold the retrieved character string.</td></tr><tr><td><i>n</i></td><td>Is the number of characters not to exceed when retrieving <i>chstr</i>.</td></tr><tr><td><i>y</i></td><td>Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved.</td></tr><tr><td><i>x</i></td><td>Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved.</td></tr></table> | <i>chstr</i> | Is a pointer to an object that can hold the retrieved character string. | <i>n</i> | Is the number of characters not to exceed when retrieving <i>chstr</i> . | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. |
| <i>chstr</i> | Is a pointer to an object that can hold the retrieved character string. | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when retrieving <i>chstr</i> . | | | | | | | | |
| <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | | | | | | | | |
| <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | | | | | | | | |

`mvinchstr(3XCURSES)`

win Is a pointer to the window in which the string is to be retrieved.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `inch(3XCURSES)`, `innstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

mvinnstr(3CURSES)

| NAME | <code> curs_instr, instr, innstr, winstr, winnstr, mvinstr, mvinnstr, mvwinstr, mvwinnstr</code> – get a string of characters from a curses window | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int instr(char *str); int innstr(char *str, int n); int winstr(WINDOW *win, char *str); int winnstr(WINDOW *win, char *str, int n); int mvinstr(int y, int x, char *str); int mvinnstr(int y, int x, char *str, int n); int mvwinstr(WINDOW *win, int y, int x, char *str); int mvwinnstr(WINDOW *win, int y, int x, char *str, int n);</pre> | | | | |
| DESCRIPTION | These routines return a string of characters in <i>str</i> , starting at the current cursor position in the named window and ending at the right margin of the window. Attributes are stripped from the characters. The four functions with <i>n</i> as the last argument return the string at most <i>n</i> characters long. | | | | |
| RETURN VALUES | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . Note that all routines except <code>winnstr()</code> may be macros. | | | | |

| | | | | | | | | | | | |
|--------------------|--|------------|---|----------|--|----------|--|----------|---|------------|--|
| NAME | innstr, instr, mvinnstr, mvinstr, mvwinnstr, mvwinstr, winstr – retrieve a multibyte character string (without rendition) | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int innstr(char *str, int n); int instr(char *str); int mvinnstr(int y, int x, char *str, int n); int mvinstr(int y, int x, char *str); int mvwinnstr(WINDOW *win, int y, int x, char *str, int n); int mvwinstr(WINDOW *win, int y, int x, char *str); int winstr(WINDOW *win, char *str); int winnstr(WINDOW *win, char *str, int n);</pre> | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>str</i></td> <td>Is a pointer to an object that can hold the retrieved multibyte character string.</td> </tr> <tr> <td><i>n</i></td> <td>Is the number of characters not to exceed when retrieving <i>str</i>.</td> </tr> <tr> <td><i>y</i></td> <td>Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved.</td> </tr> <tr> <td><i>x</i></td> <td>Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved.</td> </tr> <tr> <td><i>win</i></td> <td>Is a pointer to the window in which the string is to be retrieved.</td> </tr> </table> | <i>str</i> | Is a pointer to an object that can hold the retrieved multibyte character string. | <i>n</i> | Is the number of characters not to exceed when retrieving <i>str</i> . | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | <i>win</i> | Is a pointer to the window in which the string is to be retrieved. |
| <i>str</i> | Is a pointer to an object that can hold the retrieved multibyte character string. | | | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when retrieving <i>str</i> . | | | | | | | | | | |
| <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | | | | | | | | | | |
| <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the string is to be retrieved. | | | | | | | | | | |
| DESCRIPTION | <p>The <code>instr()</code> and <code>winstr()</code> functions retrieve a multibyte character string (without attributes) starting at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively, and ending at the right margin. The <code>mvinstr()</code> and <code>mvwinstr()</code> functions retrieve a multibyte character string located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>innstr()</code>, <code>winnstr()</code>, <code>mvinnstr()</code>, and <code>mvwinnstr()</code> functions retrieve at most <i>n</i> characters from the window <code>stdscr</code> and <i>win</i>, respectively. The former two functions retrieve the string starting at the current cursor position; the latter two commands return the string, starting at the position specified by the <i>x</i> and <i>y</i> parameters.</p> | | | | | | | | | | |

mvinnstr(3XCURSES)

All these functions store the retrieved string in the object pointed to by *str*. They only store complete multibyte characters. If the area pointed to by *str* is not large enough to hold at least one character, these functions fail.

Only the character portion of the character/rendition pair is returned. To return the complete character/rendition pair, use `winchstr()`.

ERRORS OK Successful completion.

ERR An error occurred.

USAGE All functions except `winnstr()` may be macros.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `inch(3XCURSES)`, `inchstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

NAME curs_inwstr, inwstr, innwstr, winwstr, winnwstr, mvinwstr, mvinnwstr, mvwinwstr, mvwinnwstr – get a string of wchar_t characters from a curses window

SYNOPSIS

```
cc [ flag ... ] file ... -lcurses[library .. ]
#include <curses.h>

int inwstr(wchar_t *wstr);
int innwstr(wchar_t *wstr, int n);
int winwstr(WINDOW *win, wchar_t *wstr);
int winnwstr(WINDOW *win, wchar_t *wstr, int n);
int mvinwstr(int y, int x, wchar_t *wstr);
int mvinnwstr(int y, int x, wchar_t *wstr, int n);
int mvwinwstr(WINDOW *win, int y, int x, wchar_t *wstr);
int mvwinnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);
```

DESCRIPTION These routines return the string of wchar_t characters in *wstr* starting at the current cursor position in the named window and ending at the right margin of the window. Attributes are stripped from the characters. The four functions with *n* as the last argument return the string at most *n* wchar_t characters long.

RETURN VALUES All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), attributes(5)

NOTES The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <wider.h>.

Note that all routines except winnwstr() may be macros.

mvinnwstr(3XCURSES)

| | |
|--------------------|--|
| NAME | innwstr, inwstr, mvinnwstr, mvinwstr, mvinnwstr, mvwinwstr, winnwstr, winwstr – retrieve a wide character string (without rendition) |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int innwstr(wchar_t *wstr, int n); int inwstr(wchar_t *wstr); int mvinnwstr(int y, int x, wchar_t *wstr, int n); int mvinwstr(int y, int x, wchar_t *wstr); int mvwinwstr(WINDOW*win, int y, int x, wchar_t *wstr, int n); int mvwinwstr(WINDOW*win, int y, int x, wchar_t *wstr); int winwstr(WINDOW*win, wchar_t *wstr); int winnwstr(WINDOW*win, wchar_t *wstr, int n);</pre> |
| PARAMETERS | <p><i>wstr</i> Is a pointer to an object that can hold the retrieved multibyte character string.</p> <p><i>n</i> Is the number of characters not to exceed when retrieving <i>wstr</i>.</p> <p><i>y</i> Is the y (row) coordinate of the starting position of the string to be retrieved.</p> <p><i>x</i> Is the x (column) coordinate of the starting position of the string to be retrieved.</p> <p><i>win</i> Is a pointer to the window in which the string is to be retrieved.</p> |
| DESCRIPTION | <p>The <code>inwstr()</code> and <code>winwstr()</code> functions retrieve a wide character string (without attributes) starting at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively, and ending at the right margin. The <code>mvinwstr()</code> and <code>mvwinwstr()</code> functions retrieve a wide character string located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>innwstr()</code>, <code>winnwstr()</code>, <code>mvinnwstr()</code>, and <code>mvwinwstr()</code> functions retrieve at most <i>n</i> characters from the window <code>stdscr</code> and <i>win</i>, respectively. The former two functions retrieve the string starting at the current cursor position; the latter two commands return the string, starting at the position specified by the <i>x</i> and <i>y</i> parameters.</p> <p>All these functions store the retrieved string in the object pointed to by <i>wstr</i>. They only store complete wide characters. If the area pointed to by <i>wstr</i> is not large enough to hold at least one character, these functions fail.</p> |

mvinnwstr(3XCURSES)

Only the character portion of the character/rendition pair is returned. To return the complete character/rendition pair, use `win_wchstr(3XCURSES)`.

RETURN VALUES

On success, the `inwstr()`, `mvinwstr()`, `mvwinwstr()`, and `winwstr()` functions return OK. Otherwise, they return ERR.

On success, the `innwstr()`, `mvinnwstr()`, `mvwinnwstr()`, and `winnwstr()` functions return the number of characters read into the string. Otherwise, they return ERR.

ERRORS

None.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO

`in_wch(3XCURSES)`, `in_wchnstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

mvinsch(3CURSES)

- NAME** curs_insch, insch, winsch, mvinsch, mvwinsch – insert a character before the character under the cursor in a curses window
- SYNOPSIS**

```
cc [ flag ... ] file... -lcurses [ library ... ]
#include <curses.h>

int insch(chtype ch);
int winsch(WINDOW *win, chtype ch);
int mvinsch(int y, int x, chtype ch);
int mvwinsch(WINDOW *win, int y, int x, chtype ch);
```
- DESCRIPTION** With these routines, the character *ch* is inserted before the character under the cursor. All characters to the right of the cursor are moved one space to the right, with the possibility of the rightmost character on the line being lost. The cursor position does not change (after moving to *y*, *x*, if specified). (This does not imply use of the hardware insert character feature.)
- RETURN VALUES** All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.
- ATTRIBUTES** See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

- SEE ALSO** curses(3CURSES), attributes(5)
- NOTES** The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.
- Note that `insch()`, `mvinsch()`, and `mvwinsch()` may be macros.

| NAME | insch, wunsch, mvinsch, mvwansch – insert a character | | | | | | | | |
|----------------------|--|----------------|----------------------------------|---------------------|---|----------|--|------------|--|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int insch(chtype <i>ch</i>); int mvinsch(int <i>y</i>, int <i>x</i>, chtype <i>ch</i>); int mvwansch(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>, chtype <i>ch</i>); int wansch(WINDOW *<i>win</i>, chtype <i>ch</i>);</pre> | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>ch</i></td> <td>Is the character to be inserted.</td> </tr> <tr> <td><i>y</i></td> <td>Is the y (row) coordinate of the position of the character.</td> </tr> <tr> <td><i>x</i></td> <td>Is the x (column) coordinate of the position of the character.</td> </tr> <tr> <td><i>win</i></td> <td>Is a pointer to the window in which the character is to be inserted.</td> </tr> </table> | <i>ch</i> | Is the character to be inserted. | <i>y</i> | Is the y (row) coordinate of the position of the character. | <i>x</i> | Is the x (column) coordinate of the position of the character. | <i>win</i> | Is a pointer to the window in which the character is to be inserted. |
| <i>ch</i> | Is the character to be inserted. | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of the position of the character. | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of the position of the character. | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the character is to be inserted. | | | | | | | | |
| DESCRIPTION | <p>These functions insert the character and rendition from <i>ch</i> into the current or specified window at the current or specified position.</p> <p>These functions do not perform wrapping and do not advance the cursor position. These functions perform special-character processing, with the exception that if a newline is inserted into the last line of a window and scrolling is not enabled, the behavior is unspecified.</p> | | | | | | | | |
| RETURN VALUES | Upon successful completion, these functions return OK. Otherwise, they return ERR. | | | | | | | | |
| ERRORS | No errors are defined. | | | | | | | | |
| USAGE | These functions are only guaranteed to operate reliably on character sets in which each character fits into a single byte, whose attributes can be expressed using only constants with the A_ prefix. | | | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | | | |
| Interface Stability | Standard | | | | | | | | |
| MT-Level | Unsafe | | | | | | | | |
| SEE ALSO | ins_wch(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5) | | | | | | | | |

mvinsnstr(3CURSES)

NAME | curs_Insstr, insstr, insnstr, winsstr, winsnstr, mvinsstr, mvinsnstr, mvwinsstr, mvwinsnstr – insert string before character under the cursor in a curses window

SYNOPSIS | `cc [flag ...] file ... -lcurses [library ...]
#include <curses.h>`

```
int insstr(char *str);
int insnstr(char *str, int n);
int winsstr(WINDOW *win, char *str);
int winsnstr(WINDOW *win, char *str, int n);
int mvinsstr(int y, int x, char *str);
int mvinsnstr(int y, int x, char *str, int n);
int mvwinsstr(WINDOW *win, int y, int x, char *str);
int mvwinsnstr(WINDOW *win, int y, int x, char *str, int n);
```

DESCRIPTION | With these routines, a character string (as many characters as will fit on the line) is inserted before the character under the cursor. All characters to the right of the cursor are moved to the right, with the possibility of the rightmost characters on the line being lost. The cursor position does not change (after moving to *y, x*, if specified). (This does not imply use of the hardware insert character feature.) The four routines with *n* as the last argument insert at most *n* characters. If *n* ≤ 0, then the entire string is inserted.

If a character in *str* is a tab, newline, carriage return or backspace, the cursor is moved appropriately within the window. A newline also does a `clrtoeol()` before moving. Tabs are considered to be at every eighth column. If a character in *str* is another control character, it is drawn in the `^X` notation. Calling `winch()` after adding a control character (and moving to it, if necessary) does not return the control character, but instead returns the representation of the control character.

RETURN VALUES | All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | `curs_clear(3CURSES)`, `curs_inch(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES | The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

`mvinsnstr(3COURSE)`

Note that all but `winsnstr()` may be macros.

mvinsnstr(3XCURSES)

| | | | | | | | | | | | |
|--------------------|---|------------|--|----------|--|----------|---|----------|--|------------|---|
| NAME | insnstr, insstr, mvinsnstr, mvinsstr, mvwinsnstr, mvwinsstr, winsnstr, winsstr – insert a multibyte character string | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int insnstr(const char *str, int n); int insstr(const char *str); int mvinsnstr(int y, int x, const char *str, int n); int mvinsstr(int y, int x, const char *str); int mvwinsnstr(WINDOW *win, int y, int x, const char *str, int n); int mvwinsstr(WINDOW *win, int y, int x, const char *str); int winsnstr(WINDOW *win, const char *str, int n); int winsstr(WINDOW *win, const char *str);</pre> | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>str</i></td><td>Is a pointer to the string to be inserted.</td></tr><tr><td><i>n</i></td><td>Is the number of characters not to exceed when inserting <i>str</i>. If <i>n</i> is less than 1, the entire string is inserted.</td></tr><tr><td><i>y</i></td><td>Is the y (row) coordinate of the starting position of the string.</td></tr><tr><td><i>x</i></td><td>Is the x (column) coordinate of the starting position of the string.</td></tr><tr><td><i>win</i></td><td>Is a pointer to the window in which the string is to be inserted.</td></tr></table> | <i>str</i> | Is a pointer to the string to be inserted. | <i>n</i> | Is the number of characters not to exceed when inserting <i>str</i> . If <i>n</i> is less than 1, the entire string is inserted. | <i>y</i> | Is the y (row) coordinate of the starting position of the string. | <i>x</i> | Is the x (column) coordinate of the starting position of the string. | <i>win</i> | Is a pointer to the window in which the string is to be inserted. |
| <i>str</i> | Is a pointer to the string to be inserted. | | | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when inserting <i>str</i> . If <i>n</i> is less than 1, the entire string is inserted. | | | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of the starting position of the string. | | | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of the starting position of the string. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the string is to be inserted. | | | | | | | | | | |
| DESCRIPTION | <p>The <code>insstr()</code> function inserts <i>str</i> at the current cursor position of the <code>stdscr</code> window. The <code>winsstr()</code> function performs the identical action, but in window <i>win</i>. The <code>mvinsstr()</code> and <code>mvwinsstr()</code> functions insert the character string at the starting position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <i>win</i>).</p> <p>The <code>insnstr()</code>, <code>winsnstr()</code>, <code>mvinsnstr()</code>, and <code>mvwinsnstr()</code> functions insert <i>n</i> characters to the window or as many as will fit on the line. If <i>n</i> is less than 1, the entire string is inserted or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <i>x</i> and <i>y</i> parameters.</p> <p>All characters to the right of inserted characters are moved to the right. Characters that don't fit on the current line are discarded. The cursor is left at the point of insertion.</p> | | | | | | | | | | |

mvinsnstr(3XCURSES)

If a character in *str* is a newline, carriage return, backspace, or tab, the cursor is moved appropriately. The cursor is moved to the next tab stop for each tab character (by default, tabs are eight characters apart). If the character is a control character other than those previously mentioned, the character is inserted using \hat{x} notation, where x is a printable character. `clrtoeol(3XCURSES)` is automatically done before a newline.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addchstr(3XCURSES)`, `addstr(3XCURSES)`, `clrtoeol(3XCURSES)`, `ins_nwstr(3XCURSES)`, `insch(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

mvinsnwstr(3CURSES)

| NAME | curs_inswstr, inswstr, insnwstr, winswstr, winsnwstr, mvinswstr, mvinsnwstr, mvwinswstr, mvwinsnwstr – insert wchar_t string before character under the cursor in a curses window | | | | |
|---------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int inswstr(wchar_t *wstr); int insnwstr(wchar_t *wstr, int n); int winswstr(WINDOW *win, wchar_t *wstr); int winsnwstr(WINDOW *win, wchar_t *wstr, int n); int mvinswstr(int y, int x, wchar_t *wstr); int mvinsnwstr(int y, int x, wchar_t *wstr, int n); int mvwinswstr(WINDOW *win, int y, int x, wchar_t *wstr); int mvwinsnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);</pre> | | | | |
| DESCRIPTION | <p>These routines insert a wchar_t character string (as many wchar_t characters as will fit on the line) before the character under the cursor. All characters to the right of the cursor are moved to the right, with the possibility of the rightmost characters on the line being lost. The cursor position does not change (after moving to y, x, if specified). (This does not imply use of the hardware insert character feature.) The four routines with n as the last argument insert at most n wchar_t characters. If n<=0, then the entire string is inserted.</p> <p>If a character in wstr is a tab, newline, carriage return, or backspace, the cursor is moved appropriately within the window. A newline also does a clrtoeol(3CURSES) before moving. Tabs are considered to be at every eighth column. If a character in wstr is another control character, it is drawn in the ^X notation. Calling winwch(3CURSES) after adding a control character (and moving to it, if necessary) does not return the control character, but instead returns the representation of the control character.</p> | | | | |
| RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">MT-Level</td> <td style="text-align: center;">Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | clrtoeol(3CURSES), curses(3CURSES), winwch(3CURSES), attributes(5) | | | | |
| NOTES | The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <widec.h>. | | | | |

`mvinsnWSTR(3CURSES)`

Note that all but `winsnWSTR()` may be macros.

mvins_nwstr(3XCURSES)

| | | | | | | | | | | | |
|--------------------|---|-------------|--|----------|---|----------|--|----------|---|------------|---|
| NAME | <code>ins_nwstr</code> , <code>ins_wstr</code> , <code>mvins_nwstr</code> , <code>mvins_wstr</code> , <code>mvwins_nwstr</code> , <code>mvwins_wstr</code> , <code>wins_nwstr</code> , <code>wins_wstr</code> – insert a wide character string | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int ins_nwstr(const wchar_t *wstr, int n); int ins_wstr(const wchar_t *wstr); int mvins_nwstr(int y, int x, const wchar_t *wstr, int n); int mvins_wstr(int y, int x, const wchar_t *wstr); int mvwins_nwstr(WINDOW *win, int y, int x, const wchar_t *wstr, int n); int mvwins_wstr(WINDOW *win, int y, int x, const wchar_t *wstr); int wins_nwstr(WINDOW *win, const wchar_t *wstr, int n); int wins_wstr(WINDOW *win, const wchar_t *wstr);</pre> | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>wstr</i></td><td>Is a pointer to the string to be inserted.</td></tr><tr><td><i>n</i></td><td>Is the number of characters not to exceed when inserting <i>wstr</i>. If <i>n</i> is less than 1, the entire string is inserted.</td></tr><tr><td><i>y</i></td><td>Is the <i>y</i> (row) coordinate of the starting position of the string.</td></tr><tr><td><i>x</i></td><td>Is the <i>x</i> (column) coordinate of the starting position of the string.</td></tr><tr><td><i>win</i></td><td>Is a pointer to the window in which the string is to be inserted.</td></tr></table> | <i>wstr</i> | Is a pointer to the string to be inserted. | <i>n</i> | Is the number of characters not to exceed when inserting <i>wstr</i> . If <i>n</i> is less than 1, the entire string is inserted. | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string. | <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string. | <i>win</i> | Is a pointer to the window in which the string is to be inserted. |
| <i>wstr</i> | Is a pointer to the string to be inserted. | | | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when inserting <i>wstr</i> . If <i>n</i> is less than 1, the entire string is inserted. | | | | | | | | | | |
| <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string. | | | | | | | | | | |
| <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the string is to be inserted. | | | | | | | | | | |
| DESCRIPTION | <p>The <code>ins_wstr()</code> function inserts <i>wstr</i> at the current cursor position of the <code>stdscr</code> window. The <code>wins_wstr()</code> function performs the identical action, but in window <i>win</i>. The <code>mvins_wstr()</code> and <code>mvwins_wstr()</code> functions insert <i>wstr</i> string at the starting position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>ins_nwstr()</code>, <code>wins_nwstr()</code>, <code>mvins_nwstr()</code>, and <code>mvwins_nwstr()</code> functions insert <i>n</i> characters to the window or as many as will fit on the line. If <i>n</i> is less than 1, the entire string is inserted or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <i>x</i> and <i>y</i> parameters.</p> | | | | | | | | | | |

mvins_nwstr(3XCURSES)

All characters to the right of inserted characters are moved to the right. Characters that don't fit on the current line are discarded. The cursor is left at the point of insertion.

If a character in *wstr* is a newline, carriage return, backspace, or tab, the cursor is moved appropriately. The cursor is moved to the next tab stop for each tab character (by default, tabs are eight characters apart). If the character is a control character other than those previously mentioned, the character is inserted using \hat{x} notation, where x is a printable character. `clrtoeol(3XCURSES)` is automatically done before a newline.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `add_wchnstr(3XCURSES)`, `addnwstr(3XCURSES)`, `clrtoeol(3XCURSES)`, `ins_wch(3XCURSES)`, `insnstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

mvinsstr(3CURSES)

| NAME | <code> curs_ insstr, insstr, insnstr, winsstr, winsnstr, mvinsstr, mvinsnstr, mvwinsstr, mvwinsnstr </code> – insert string before character under the cursor in a curses window | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int insstr(char *str); int insnstr(char *str, int n); int winsstr(WINDOW *win, char *str); int winsnstr(WINDOW *win, char *str, int n); int mvinsstr(int y, int x, char *str); int mvinsnstr(int y, int x, char *str, int n); int mvwinsstr(WINDOW *win, int y, int x, char *str); int mvwinsnstr(WINDOW *win, int y, int x, char *str, int n);</pre> | | | | |
| DESCRIPTION | <p>With these routines, a character string (as many characters as will fit on the line) is inserted before the character under the cursor. All characters to the right of the cursor are moved to the right, with the possibility of the rightmost characters on the line being lost. The cursor position does not change (after moving to <i>y, x</i>, if specified). (This does not imply use of the hardware insert character feature.) The four routines with <i>n</i> as the last argument insert at most <i>n</i> characters. If <i>n</i> ≤ 0, then the entire string is inserted.</p> <p>If a character in <i>str</i> is a tab, newline, carriage return or backspace, the cursor is moved appropriately within the window. A newline also does a <code>clrtoeol()</code> before moving. Tabs are considered to be at every eighth column. If a character in <i>str</i> is another control character, it is drawn in the <code>^X</code> notation. Calling <code>winch()</code> after adding a control character (and moving to it, if necessary) does not return the control character, but instead returns the representation of the control character.</p> | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code> curs_ clear(3CURSES), curs_ inch(3CURSES), curses(3CURSES), attributes(5) </code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . | | | | |

`mvinsstr(3CURSES)`

Note that all but `winsnstr()` may be macros.

mvinsstr(3XCURSES)

| | | | | | | | | | | | |
|--------------------|---|------------|--|----------|--|----------|---|----------|--|------------|---|
| NAME | insnstr, insstr, mvinsnstr, mvinsstr, mvwinsnstr, mvwinsstr, winsnstr, winsstr – insert a multibyte character string | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int insnstr(const char *str, int n); int insstr(const char *str); int mvinsnstr(int y, int x, const char *str, int n); int mvinsstr(int y, int x, const char *str); int mvwinsnstr(WINDOW *win, int y, int x, const char *str, int n); int mvwinsstr(WINDOW *win, int y, int x, const char *str); int winsnstr(WINDOW *win, const char *str, int n); int winsstr(WINDOW *win, const char *str);</pre> | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>str</i></td><td>Is a pointer to the string to be inserted.</td></tr><tr><td><i>n</i></td><td>Is the number of characters not to exceed when inserting <i>str</i>. If <i>n</i> is less than 1, the entire string is inserted.</td></tr><tr><td><i>y</i></td><td>Is the y (row) coordinate of the starting position of the string.</td></tr><tr><td><i>x</i></td><td>Is the x (column) coordinate of the starting position of the string.</td></tr><tr><td><i>win</i></td><td>Is a pointer to the window in which the string is to be inserted.</td></tr></table> | <i>str</i> | Is a pointer to the string to be inserted. | <i>n</i> | Is the number of characters not to exceed when inserting <i>str</i> . If <i>n</i> is less than 1, the entire string is inserted. | <i>y</i> | Is the y (row) coordinate of the starting position of the string. | <i>x</i> | Is the x (column) coordinate of the starting position of the string. | <i>win</i> | Is a pointer to the window in which the string is to be inserted. |
| <i>str</i> | Is a pointer to the string to be inserted. | | | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when inserting <i>str</i> . If <i>n</i> is less than 1, the entire string is inserted. | | | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of the starting position of the string. | | | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of the starting position of the string. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the string is to be inserted. | | | | | | | | | | |
| DESCRIPTION | <p>The <code>insstr()</code> function inserts <i>str</i> at the current cursor position of the <code>stdscr</code> window. The <code>winsstr()</code> function performs the identical action, but in window <i>win</i>. The <code>mvinsstr()</code> and <code>mvwinsstr()</code> functions insert the character string at the starting position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <i>win</i>).</p> <p>The <code>insnstr()</code>, <code>winsnstr()</code>, <code>mvinsnstr()</code>, and <code>mvwinsnstr()</code> functions insert <i>n</i> characters to the window or as many as will fit on the line. If <i>n</i> is less than 1, the entire string is inserted or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <i>x</i> and <i>y</i> parameters.</p> <p>All characters to the right of inserted characters are moved to the right. Characters that don't fit on the current line are discarded. The cursor is left at the point of insertion.</p> | | | | | | | | | | |

`mvinsstr(3XCURSES)`

If a character in *str* is a newline, carriage return, backspace, or tab, the cursor is moved appropriately. The cursor is moved to the next tab stop for each tab character (by default, tabs are eight characters apart). If the character is a control character other than those previously mentioned, the character is inserted using \hat{x} notation, where *x* is a printable character. `clrtoeol(3XCURSES)` is automatically done before a newline.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addchstr(3XCURSES)`, `addstr(3XCURSES)`, `clrtoeol(3XCURSES)`, `ins_nwstr(3XCURSES)`, `insch(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

mvinstr(3CURSES)

| NAME | <code> curs_instr, instr, innstr, winstr, winnstr, mvinstr, mvinnstr, mvwinstr, mvwinnstr</code> – get a string of characters from a curses window | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int instr(char *str); int innstr(char *str, int n); int winstr(WINDOW *win, char *str); int winnstr(WINDOW *win, char *str, int n); int mvinstr(int y, int x, char *str); int mvinnstr(int y, int x, char *str, int n); int mvwinstr(WINDOW *win, int y, int x, char *str); int mvwinnstr(WINDOW *win, int y, int x, char *str, int n);</pre> | | | | |
| DESCRIPTION | These routines return a string of characters in <i>str</i> , starting at the current cursor position in the named window and ending at the right margin of the window. Attributes are stripped from the characters. The four functions with <i>n</i> as the last argument return the string at most <i>n</i> characters long. | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . Note that all routines except <code>winnstr()</code> may be macros. | | | | |

| | | | | | | | | | | | |
|--------------------|---|------------|---|----------|--|----------|--|----------|---|------------|--|
| NAME | innstr, instr, mvinnstr, mvinstr, mvwinstr, mvwinstr, winstr – retrieve a multibyte character string (without rendition) | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int innstr(char *str, int n); int instr(char *str); int mvinnstr(int y, int x, char *str, int n); int mvinstr(int y, int x, char *str); int mvwinstr(WINDOW *win, int y, int x, char *str, int n); int mvwinstr(WINDOW *win, int y, int x, char *str); int winstr(WINDOW *win, char *str); int winstr(WINDOW *win, char *str, int n);</pre> | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>str</i></td> <td>Is a pointer to an object that can hold the retrieved multibyte character string.</td> </tr> <tr> <td><i>n</i></td> <td>Is the number of characters not to exceed when retrieving <i>str</i>.</td> </tr> <tr> <td><i>y</i></td> <td>Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved.</td> </tr> <tr> <td><i>x</i></td> <td>Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved.</td> </tr> <tr> <td><i>win</i></td> <td>Is a pointer to the window in which the string is to be retrieved.</td> </tr> </table> | <i>str</i> | Is a pointer to an object that can hold the retrieved multibyte character string. | <i>n</i> | Is the number of characters not to exceed when retrieving <i>str</i> . | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | <i>win</i> | Is a pointer to the window in which the string is to be retrieved. |
| <i>str</i> | Is a pointer to an object that can hold the retrieved multibyte character string. | | | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when retrieving <i>str</i> . | | | | | | | | | | |
| <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | | | | | | | | | | |
| <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the string is to be retrieved. | | | | | | | | | | |
| DESCRIPTION | <p>The <code>instr()</code> and <code>winstr()</code> functions retrieve a multibyte character string (without attributes) starting at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively, and ending at the right margin. The <code>mvinstr()</code> and <code>mvwinstr()</code> functions retrieve a multibyte character string located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>innstr()</code>, <code>winnstr()</code>, <code>mvinnstr()</code>, and <code>mvwinstr()</code> functions retrieve at most <i>n</i> characters from the window <code>stdscr</code> and <i>win</i>, respectively. The former two functions retrieve the string starting at the current cursor position; the latter two commands return the string, starting at the position specified by the <i>x</i> and <i>y</i> parameters.</p> | | | | | | | | | | |

`mvinstr(3XCURSES)`

All these functions store the retrieved string in the object pointed to by *str*. They only store complete multibyte characters. If the area pointed to by *str* is not large enough to hold at least one character, these functions fail.

Only the character portion of the character/rendition pair is returned. To return the complete character/rendition pair, use `winchstr()`.

ERRORS OK Successful completion.

ERR An error occurred.

USAGE All functions except `winnstr()` may be macros.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `inch(3XCURSES)`, `inchstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_inswch, inswch, winswch, mvinswch, mvwinswch – insert a wchar_t character before the character under the cursor in a curses window | | | | |
|---------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int inswch(chtype wch); int winswch(WINDOW *win, chtype wch); int mvinswch(int y, int x, chtype wch); int mvwinswch(WINDOW *win, int y, int x, chtype wch);</pre> | | | | |
| DESCRIPTION | <p>These routines insert the character <i>wch</i>, holding a wchar_t character, before the character under the cursor. All characters to the right of the cursor are moved one space to the right, with the possibility of the rightmost character on the line being lost. The cursor position does not change (after moving to <i>y, x</i>, if specified). (This does not imply use of the hardware insert character feature.)</p> | | | | |
| RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">MT-Level</td> <td style="text-align: center;">Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), attributes(5) | | | | |
| NOTES | <p>The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <widec.h>.</p> <p>Note that inswch(), mvinswch(), and mvwinswch() may be macros.</p> <p>None of these routines can use the color attribute in chtype.</p> | | | | |

mvins_wch(3XCURSES)

NAME | ins_wch, wins_wch, mvins_wch, mvwins_wch – insert a complex character

SYNOPSIS | `cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \`
`-R /usr/xpg4/lib -lcurses [library...]`

c89 | `[flag...] file... -lcurses [library...]`

```
#include <curses.h>
int ins_wch(const cchar_t *wch);
int mvins_wch(int y, int x, const cchar_t *wch);
int mvwins_wch(WINDOW *win, int y, int x, const cchar_t *wch);
int wins_wch(WINDOW *win, const cchar_t *wch);
```

PARAMETERS

| | |
|------------|--|
| <i>wch</i> | Is the complex character to be inserted. |
| <i>y</i> | Is the y (row) coordinate of the position of the character. |
| <i>x</i> | Is the x (column) coordinate of the position of the character. |
| <i>win</i> | Is a pointer to the window in which the character is to be inserted. |

DESCRIPTION

The `ins_wch()` function inserts the complex character *wch* at the current cursor position of the `stdscr` window. The `wins_wch()` function performs the identical action but in window *win*. The `mvins_wch()` and `mvwins_wch()` functions insert the character at the position indicated by the *x* (column) and *y* (row) parameters (the former in the `stdscr` window; the latter in window *win*). The cursor position does not change.

All characters to the right of the inserted character are moved right one character. The last character on the line is deleted.

Insertions and deletions occur at the character level. The cursor is adjusted to the first column of the character prior to the the operation.

RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR.

ERRORS | None.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

`mvins_wch(3XCURSES)`

SEE ALSO `add_wch(3XCURSES)`, `ins_nwstr(3XCURSES)`, `libcurses(3XCURSES)`,
`attributes(5)`, `standards(5)`

mvinswstr(3CURSES)

| NAME | curs_inswstr, inswstr, insnwstr, winswstr, winsnwstr, mvinswstr, mvinsnwstr, mvwinswstr, mvwinsnwstr – insert wchar_t string before character under the cursor in a curses window | | | | |
|---------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int inswstr(wchar_t *wstr); int insnwstr(wchar_t *wstr, int n); int winswstr(WINDOW *win, wchar_t *wstr); int winsnwstr(WINDOW *win, wchar_t *wstr, int n); int mvinswstr(int y, int x, wchar_t *wstr); int mvinsnwstr(int y, int x, wchar_t *wstr, int n); int mvwinswstr(WINDOW *win, int y, int x, wchar_t *wstr); int mvwinsnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);</pre> | | | | |
| DESCRIPTION | <p>These routines insert a wchar_t character string (as many wchar_t characters as will fit on the line) before the character under the cursor. All characters to the right of the cursor are moved to the right, with the possibility of the rightmost characters on the line being lost. The cursor position does not change (after moving to y, x, if specified). (This does not imply use of the hardware insert character feature.) The four routines with n as the last argument insert at most n wchar_t characters. If n<=0, then the entire string is inserted.</p> <p>If a character in wstr is a tab, newline, carriage return, or backspace, the cursor is moved appropriately within the window. A newline also does a clrtoeol(3CURSES) before moving. Tabs are considered to be at every eighth column. If a character in wstr is another control character, it is drawn in the ^X notation. Calling winwch(3CURSES) after adding a control character (and moving to it, if necessary) does not return the control character, but instead returns the representation of the control character.</p> | | | | |
| RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | clrtoeol(3CURSES), curses(3CURSES), winwch(3CURSES), attributes(5) | | | | |
| NOTES | The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <widec.h>. | | | | |

`mvinswstr(3CURSES)`

Note that all but `winswstr()` may be macros.

mvins_wstr(3XCURSES)

| | | | | | | | | | | | |
|--------------------|---|-------------|--|----------|---|----------|---|----------|--|------------|---|
| NAME | <code>ins_nwstr</code> , <code>ins_wstr</code> , <code>mvins_nwstr</code> , <code>mvins_wstr</code> , <code>mvwins_nwstr</code> , <code>mvwins_wstr</code> , <code>wins_nwstr</code> , <code>wins_wstr</code> – insert a wide character string | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int ins_nwstr(const wchar_t *wstr, int n); int ins_wstr(const wchar_t *wstr); int mvins_nwstr(int y, int x, const wchar_t *wstr, int n); int mvins_wstr(int y, int x, const wchar_t *wstr); int mvwins_nwstr(WINDOW *win, int y, int x, const wchar_t *wstr, int n); int mvwins_wstr(WINDOW *win, int y, int x, const wchar_t *wstr); int wins_nwstr(WINDOW *win, const wchar_t *wstr, int n); int wins_wstr(WINDOW *win, const wchar_t *wstr);</pre> | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>wstr</i></td><td>Is a pointer to the string to be inserted.</td></tr><tr><td><i>n</i></td><td>Is the number of characters not to exceed when inserting <i>wstr</i>. If <i>n</i> is less than 1, the entire string is inserted.</td></tr><tr><td><i>y</i></td><td>Is the y (row) coordinate of the starting position of the string.</td></tr><tr><td><i>x</i></td><td>Is the x (column) coordinate of the starting position of the string.</td></tr><tr><td><i>win</i></td><td>Is a pointer to the window in which the string is to be inserted.</td></tr></table> | <i>wstr</i> | Is a pointer to the string to be inserted. | <i>n</i> | Is the number of characters not to exceed when inserting <i>wstr</i> . If <i>n</i> is less than 1, the entire string is inserted. | <i>y</i> | Is the y (row) coordinate of the starting position of the string. | <i>x</i> | Is the x (column) coordinate of the starting position of the string. | <i>win</i> | Is a pointer to the window in which the string is to be inserted. |
| <i>wstr</i> | Is a pointer to the string to be inserted. | | | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when inserting <i>wstr</i> . If <i>n</i> is less than 1, the entire string is inserted. | | | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of the starting position of the string. | | | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of the starting position of the string. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the string is to be inserted. | | | | | | | | | | |
| DESCRIPTION | <p>The <code>ins_wstr()</code> function inserts <i>wstr</i> at the current cursor position of the <code>stdscr</code> window. The <code>wins_wstr()</code> function performs the identical action, but in window <i>win</i>. The <code>mvins_wstr()</code> and <code>mvwins_wstr()</code> functions insert <i>wstr</i> string at the starting position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>ins_nwstr()</code>, <code>wins_nwstr()</code>, <code>mvins_nwstr()</code>, and <code>mvwins_nwstr()</code> functions insert <i>n</i> characters to the window or as many as will fit on the line. If <i>n</i> is less than 1, the entire string is inserted or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <i>x</i> and <i>y</i> parameters.</p> | | | | | | | | | | |

`mvins_wstr(3XCURSES)`

All characters to the right of inserted characters are moved to the right. Characters that don't fit on the current line are discarded. The cursor is left at the point of insertion.

If a character in *wstr* is a newline, carriage return, backspace, or tab, the cursor is moved appropriately. The cursor is moved to the next tab stop for each tab character (by default, tabs are eight characters apart). If the character is a control character other than those previously mentioned, the character is inserted using \hat{x} notation, where *x* is a printable character. `clrtoeol(3XCURSES)` is automatically done before a newline.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `add_wchnstr(3XCURSES)`, `addnwstr(3XCURSES)`, `clrtoeol(3XCURSES)`, `ins_wch(3XCURSES)`, `insnstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

mvinwch(3CURSES)

NAME curs_inwch, inwch, winwch, mvinwch, mvwinwch – get a wchar_t character and its attributes from a curses window

SYNOPSIS cc [*flag* ...] *file* ... -lcurses [*library* ..]
#include <curses.h>

```
ctype inwch(void);  
ctype winwch(WINDOW *win);  
ctype mvinwch(int y, int x);  
ctype mvwinwch(WINDOW *win, int y, int x);
```

DESCRIPTION These routines return the wchar_t character, of type ctype, at the current position in the named window. If any attributes are set for that position, their values are OR-ed into the value returned. Constants defined in <curses.h> can be used with the logical AND (&) operator to extract the character or attributes alone.

Attributes The following bit-masks may be AND-ed with characters returned by winwch().

```
A_WCHARTEXT      Bit-mask to extract character  
A_WATTRIBUTES   Bit-mask to extract attributes
```

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), attributes(5)

NOTES The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <wdec.h>.

Note that all of these routines may be macros.

None of these routines can use the color attribute in ctype.

NAME in_wch, mvin_wch, mvwin_wch, win_wch – retrieve a complex character (with rendition)

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

int in_wch(cchar_t *wcv);

int mvin_wch(int y, int x, cchar_t *wcv);

int mvwin_wch(WINDOW *win, int y, cchar_t *wcv);

int win_wch(WINDOW *win, cchar_t *wcv);
```

DESCRIPTION The in_wch() and win_wch() functions retrieve the complex character and its rendition located at the current cursor position of the stdscr window and window win, respectively. The mvin_wch() and mvwin_wch() functions retrieve the complex character and its rendition located at the position indicated by the x (column) and y (row) parameters (the former in the stdscr window; the latter in window win).

All these functions store the retrieved character and its rendition in the object pointed to by wcv.

PARAMETERS

| | |
|-----|---|
| wcv | Is a pointer to an object that can store a complex character and its rendition. |
| y | Is the y (row) coordinate of the position of the character to be returned. |
| x | Is the x (column) coordinate of the position of the character to be returned. |
| win | Is a pointer to the window that contains the character to be returned. |

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO add_wch(3XCURSES), inch(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

mvinwchnstr(3CURSES)

| NAME | curs_inwchstr, inwchstr, inwchnstr, winwchstr, winwchnstr, mvinwchstr, mvinwchnstr, mvwinwchstr, mvwinwchnstr – get a string of wchar_t characters (and attributes) from a curses window | | | | |
|---------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int inwchstr (chtype *wchstr); int inwchnstr (chtype *wchstr, int n); int winwchstr (WINDOW *win, chtype *wchstr); int winwchnstr (WINDOW *win, chtype *wchstr, int n); int mvinwchstr (int y, int x, chtype *wchstr); int mvinwchnstr (int y, int x, chtype *wchstr, int n); int mvwinwchstr (WINDOW *win, int y, int x, chtype *wchstr); int mvwinwchnstr (WINDOW *win, int y, int x, chtype *wchstr, int n);</pre> | | | | |
| DESCRIPTION | These routines return a string of type chtype, holding wchar_t characters, starting at the current cursor position in the named window and ending at the right margin of the window. The four functions with <i>n</i> as the last argument, return the string at most <i>n</i> wchar_t characters long. Constants defined in <curses.h> can be used with the logical AND (&) operator to extract the wchar_t character or the attribute alone from any position in the wchstr (see curs_inwch(3CURSES)). | | | | |
| RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See attributes(5) for a description of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">MT-Level</td> <td style="text-align: center;">Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), curs_inwch(3CURSES), attributes(5) | | | | |
| NOTES | <p>The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <wider.h>.</p> <p>Note that all routines except winwchnstr() may be macros.</p> <p>None of these routines can use the color attribute in chtype.</p> | | | | |

| | | | | | | | | | | | |
|--------------------|--|---------------|---|----------|---|----------|---|----------|--|------------|--|
| NAME | in_wchnstr, in_wchstr, mvin_wchnstr, mvin_wchstr, mvwin_wchnstr, mvwin_wchstr, win_wchnstr, win_wchstr – retrieve complex character string (with rendition) | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int in_wchnstr(cchar_t *wchstr, int n); int in_wchstr(cchar_t *wchstr); int mvin_wchnstr(int y, int x, cchar_t *wchstr, int n); int mvin_wchstr(int y, int x, cchar_t *wchstr); int mvwin_wchnstr(WINDOW *win, int y, int x, cchar_t *wchstr, int n); int mvwin_wchstr(WINDOW *win, int y, int x, cchar_t *wchstr); int win_wchnstr(WINDOW *win, cchar_t *wchstr, int n); int win_wchstr(WINDOW *win, cchar_t *wchstr);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The in_wchstr() and win_wchstr() functions retrieve a complex character string (with rendition) starting at the current cursor position of the stdscr window and window win, respectively, and ending at the right margin. The mvin_wchstr() and mvwin_wchstr() functions retrieve a complex character string located at the position indicated by the x (column) and y (row) parameters (the former in the stdscr window; the latter in window win).</p> <p>The in_wchnstr(), win_wchnstr(), mvin_wchnstr(), and mvwin_wchnstr() functions retrieve at most n characters from the window stdscr and win, respectively. The former two functions retrieve the string, starting at the current cursor position; the latter two commands retrieve the string, starting at the position specified by the x and y parameters.</p> <p>The retrieved character string (with renditions) is stored in the object pointed to by wccval.</p> | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>wchstr</i></td> <td>Is a pointer to an object where the retrieved complex character string can be stored.</td> </tr> <tr> <td><i>n</i></td> <td>Is the number of characters not to exceed when retrieving <i>wchstr</i>.</td> </tr> <tr> <td><i>y</i></td> <td>Is the y (row) coordinate of the starting position of the string to be retrieved.</td> </tr> <tr> <td><i>x</i></td> <td>Is the x (column) coordinate of the starting position of the string to be retrieved.</td> </tr> <tr> <td><i>win</i></td> <td>Is a pointer to the window in which the string is to be retrieved.</td> </tr> </table> | <i>wchstr</i> | Is a pointer to an object where the retrieved complex character string can be stored. | <i>n</i> | Is the number of characters not to exceed when retrieving <i>wchstr</i> . | <i>y</i> | Is the y (row) coordinate of the starting position of the string to be retrieved. | <i>x</i> | Is the x (column) coordinate of the starting position of the string to be retrieved. | <i>win</i> | Is a pointer to the window in which the string is to be retrieved. |
| <i>wchstr</i> | Is a pointer to an object where the retrieved complex character string can be stored. | | | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when retrieving <i>wchstr</i> . | | | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of the starting position of the string to be retrieved. | | | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of the starting position of the string to be retrieved. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the string is to be retrieved. | | | | | | | | | | |

`mvin_wchnstr(3XCURSES)`

RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR.

ERRORS | None.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | `in_wch(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

NAME | curs_inwchstr, inwchstr, inwchnstr, winwchstr, winwchnstr, mvinwchstr, mvinwchnstr, mvwinwchstr, mvwinwchnstr – get a string of wchar_t characters (and attributes) from a curses window

SYNOPSIS | `cc [flag ...] file ... -lcurses [library ..]
#include <curses.h>
int inwchstr(chtype *wchstr);
int inwchnstr(chtype *wchstr, int n);
int winwchstr(WINDOW *win, chtype *wchstr);
int winwchnstr(WINDOW *win, chtype *wchstr, int n);
int mvinwchstr(int y, int x, chtype *wchstr);
int mvinwchnstr(int y, int x, chtype *wchstr, int n);
int mvwinwchstr(WINDOW *win, int y, int x, chtype *wchstr);
int mvwinwchnstr(WINDOW *win, int y, int x, chtype *wchstr, int n);`

DESCRIPTION | These routines return a string of type chtype, holding wchar_t characters, starting at the current cursor position in the named window and ending at the right margin of the window. The four functions with *n* as the last argument, return the string at most *n* wchar_t characters long. Constants defined in <curses.h> can be used with the logical AND (&) operator to extract the wchar_t character or the attribute alone from any position in the *wchstr* (see curs_inwch(3CURSES)).

RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES | See attributes(5) for a description of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | curses(3CURSES), curs_inwch(3CURSES), attributes(5)

NOTES | The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <widec.h>.

Note that all routines except winwchnstr() may be macros.

None of these routines can use the color attribute in chtype.

mvin_wchstr(3XCURSES)

| | | | | | | | | | | | |
|--------------------|---|---------------|---|----------|---|----------|--|----------|---|------------|--|
| NAME | in_wchnstr, in_wchstr, mvin_wchnstr, mvin_wchstr, mvwin_wchnstr, mvwin_wchstr, win_wchnstr, win_wchstr – retrieve complex character string (with rendition) | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int in_wchnstr(cchar_t *wchstr, int n); int in_wchstr(cchar_t *wchstr); int mvin_wchnstr(int y, int x, cchar_t *wchstr, int n); int mvin_wchstr(int y, int x, cchar_t *wchstr); int mvwin_wchnstr(WINDOW *win, int y, int x, cchar_t *wchstr, int n); int mvwin_wchstr(WINDOW *win, int y, int x, cchar_t *wchstr); int win_wchnstr(WINDOW *win, cchar_t *wchstr, int n); int win_wchstr(WINDOW *win, cchar_t *wchstr);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The <code>in_wchstr()</code> and <code>win_wchstr()</code> functions retrieve a complex character string (with rendition) starting at the current cursor position of the <code>stdscr</code> window and window <code>win</code>, respectively, and ending at the right margin. The <code>mvin_wchstr()</code> and <code>mvwin_wchstr()</code> functions retrieve a complex character string located at the position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <code>win</code>).</p> <p>The <code>in_wchnstr()</code>, <code>win_wchnstr()</code>, <code>mvin_wchnstr()</code>, and <code>mvwin_wchnstr()</code> functions retrieve at most <code>n</code> characters from the window <code>stdscr</code> and <code>win</code>, respectively. The former two functions retrieve the string, starting at the current cursor position; the latter two commands retrieve the string, starting at the position specified by the <code>x</code> and <code>y</code> parameters.</p> <p>The retrieved character string (with renditions) is stored in the object pointed to by <code>wcval</code>.</p> | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>wchstr</i></td> <td>Is a pointer to an object where the retrieved complex character string can be stored.</td> </tr> <tr> <td><i>n</i></td> <td>Is the number of characters not to exceed when retrieving <i>wchstr</i>.</td> </tr> <tr> <td><i>y</i></td> <td>Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved.</td> </tr> <tr> <td><i>x</i></td> <td>Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved.</td> </tr> <tr> <td><i>win</i></td> <td>Is a pointer to the window in which the string is to be retrieved.</td> </tr> </table> | <i>wchstr</i> | Is a pointer to an object where the retrieved complex character string can be stored. | <i>n</i> | Is the number of characters not to exceed when retrieving <i>wchstr</i> . | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | <i>win</i> | Is a pointer to the window in which the string is to be retrieved. |
| <i>wchstr</i> | Is a pointer to an object where the retrieved complex character string can be stored. | | | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when retrieving <i>wchstr</i> . | | | | | | | | | | |
| <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | | | | | | | | | | |
| <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the string is to be retrieved. | | | | | | | | | | |

`mvin_wchstr(3XCURSES)`

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `in_wch(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

mvinwstr(3CURSES)

| NAME | <code> curs_inwstr, inwstr, innwstr, winwstr, winnwstr, mvinwstr, mvinnwstr, mvwinwstr, mvwinnwstr</code> – get a string of <code>wchar_t</code> characters from a curses window | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses[library ..] #include <curses.h> int inwstr(wchar_t *wstr); int innwstr(wchar_t *wstr, int n); int winwstr(WINDOW *win, wchar_t *wstr); int winnwstr(WINDOW *win, wchar_t *wstr, int n); int mvinwstr(int y, int x, wchar_t *wstr); int mvinnwstr(int y, int x, wchar_t *wstr, int n); int mvwinwstr(WINDOW *win, int y, int x, wchar_t *wstr); int mvwinnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);</pre> | | | | |
| DESCRIPTION | These routines return the string of <code>wchar_t</code> characters in <code>wstr</code> starting at the current cursor position in the named window and ending at the right margin of the window. Attributes are stripped from the characters. The four functions with <code>n</code> as the last argument return the string at most <code>n</code> <code>wchar_t</code> characters long. | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header file <code><curses.h></code> automatically includes the header files <code><stdio.h></code> , <code><unctrl.h></code> and <code><wctype.h></code> . Note that all routines except <code>winnwstr()</code> may be macros. | | | | |

| | |
|--------------------|---|
| NAME | innwstr, inwstr, mvinnwstr, mvinwstr, mvwinnwstr, mvwinwstr, winnwstr, winwstr – retrieve a wide character string (without rendition) |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int innwstr(wchar_t *wstr, int n); int inwstr(wchar_t *wstr); int mvinnwstr(int y, int x, wchar_t *wstr, int n); int mvinwstr(int y, int x, wchar_t *wstr); int mvwinnwstr(WINDOW*win, int y, int x, wchar_t *wstr, int n); int mvwinwstr(WINDOW*win, int y, int x, wchar_t *wstr); int winwstr(WINDOW*win, wchar_t *wstr); int winnwstr(WINDOW*win, wchar_t *wstr, int n);</pre> |
| PARAMETERS | <p><i>wstr</i> Is a pointer to an object that can hold the retrieved multibyte character string.</p> <p><i>n</i> Is the number of characters not to exceed when retrieving <i>wstr</i>.</p> <p><i>y</i> Is the y (row) coordinate of the starting position of the string to be retrieved.</p> <p><i>x</i> Is the x (column) coordinate of the starting position of the string to be retrieved.</p> <p><i>win</i> Is a pointer to the window in which the string is to be retrieved.</p> |
| DESCRIPTION | <p>The <code>inwstr()</code> and <code>winwstr()</code> functions retrieve a wide character string (without attributes) starting at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively, and ending at the right margin. The <code>mvinwstr()</code> and <code>mvwinwstr()</code> functions retrieve a wide character string located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>innwstr()</code>, <code>winnwstr()</code>, <code>mvinnwstr()</code>, and <code>mvwinnwstr()</code> functions retrieve at most <i>n</i> characters from the window <code>stdscr</code> and <i>win</i>, respectively. The former two functions retrieve the string starting at the current cursor position; the latter two commands return the string, starting at the position specified by the <i>x</i> and <i>y</i> parameters.</p> <p>All these functions store the retrieved string in the object pointed to by <i>wstr</i>. They only store complete wide characters. If the area pointed to by <i>wstr</i> is not large enough to hold at least one character, these functions fail.</p> |

`mvinwstr(3XCURSES)`

Only the character portion of the character/rendition pair is returned. To return the complete character/rendition pair, use `win_wchstr(3XCURSES)`.

RETURN VALUES On success, the `inwstr()`, `mvinwstr()`, `mvwinwstr()`, and `winwstr()` functions return OK. Otherwise, they return ERR.

On success, the `innwstr()`, `mvinnwstr()`, `mvwinnwstr()`, and `winnwstr()` functions return the number of characters read into the string. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `in_wch(3XCURSES)`, `in_wchnstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_printw, printw, wprintw, mvprintw, mvwprintw, vwprintw – print formatted output in curses windows | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int printw(char *fmt, /* arg */ ...); int wprintw(WINDOW *win, char *fmt, /* arg */ ...); int mvprintw(int y, int x, char *fmt, /* arg */ ...); int mvwprintw(WINDOW *win, int y, int x, char *fmt, /* arg */...); #include <varargs.h> int vwprintw(WINDOW *win, char *fmt, /* varlist */ ...);</pre> | | | | |
| DESCRIPTION | <p>The <code>printw()</code>, <code>wprintw()</code>, <code>mvprintw()</code>, and <code>mvwprintw()</code> routines are analogous to <code>printf()</code> (see <code>printf(3C)</code>). In effect, the string that would be output by <code>printf()</code> is output instead as though <code>waddstr()</code> were used on the given window.</p> <p>The <code>vwprintw()</code> routine is analogous to <code>vprintf()</code> (see <code>vprintf(3C)</code>) and performs a <code>wprintw()</code> using a variable argument list. The third argument is a <code>va_list</code>, a pointer to a list of arguments, as defined in <code><varargs.h></code>.</p> | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>printf(3C)</code> , <code>vprintf(3C)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . | | | | |

mvprintw(3XCURSES)

| NAME | mvprintw, mvwprintw, printw, wprintw – print formatted output window | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int mvprintw(int y, int x, char *fmt, ...); int mvwprintw(WINDOW *win, int y, int x, char *fmt, ...); int printw(char *fmt, ...); int wprintw(WINDOW *win, char *fmt, ...);</pre> | | | | | | |
| PARAMETERS | <p><i>y</i> Is the y (row) coordinate position of the string's placement in the window.</p> <p><i>x</i> Is the x (column) coordinate position of the string's placement in the window.</p> <p><i>fmt</i> Is a printf() format string.</p> <p><i>win</i> Is a pointer to the window in which the string is to be written.</p> | | | | | | |
| DESCRIPTION | The mvprintw(), mvwprintw(), printw(), and wprintw() functions are analogous to printf(3C). The effect of these functions is as though sprintf() were used to format the string, and then waddstr(3XCURSES) were used to add that multi-byte string to the current or specified window at the current or specified cursor position. | | | | | | |
| RETURN VALUES | Upon successful completion, these functions return OK. Otherwise, they return ERR. | | | | | | |
| ERRORS | No errors are defined. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | addnstr(3XCURSES), libcurses(3XCURSES), printf(3C), attributes(5), standards(5) | | | | | | |

| NAME | curs_scanw, scanw, wscanw, mvscanw, mvwscanw, vwscanw – convert formatted input from a curses widow | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int scanw(char *fmt, /* arg */ ...); int wscanw(WINDOW *win, char *fmt, /* arg */ ...); int mvscanw(int y, int x, char *fmt, /* arg */ ...); int mvwscanw(WINDOW *win, int y, int x, char *fmt, /* arg */...); int vwscanw(WINDOW *win, char *fmt, va_list varlist);</pre> | | | | |
| DESCRIPTION | <p>The scanw(), wscanw(), and mvscanw() routines correspond to scanf() (see scanf(3C)). The effect of these routines is as though wgetstr() were called on the window, and the resulting line used as input for the scan. Fields which do not map to a variable in the fmt field are lost.</p> <p>The vwscanw() routine is similar to vwprintw() in that it performs a wscanw() using a variable argument list. The third argument is a va_list, a pointer to a list of arguments, as defined in <varargs.h>.</p> | | | | |
| RETURN VALUES | <p>vwscanw() returns ERR on failure and an integer equal to the number of fields scanned on success.</p> <p>Applications may interrogate the return value from the scanw, wscanw(), mvscanw(), and mvwscanw() routines to determine the number of fields which were mapped in the call.</p> | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curs_getstr(3CURSES), curs_printw(3CURSES), curses(3CURSES), scanf(3C), attributes(5) | | | | |
| NOTES | The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>. | | | | |

mvscanw(3XCURSES)

NAME mvscanw, mvwscanw, scanw, wscanw – convert formatted input from a window

SYNOPSIS `cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [library...]`

`c89 [flag...] file... -lcurses [library...]`

```
#include <curses.h>
int mvscanw(int y, int x, char *fmt, ...);
int mvwscanw(WINDOW *win, int y, int x, char *fmt, ...);
int scanw(char *fmt, ...);
int wscanw(WINDOW *win, char *fmt, ...);
```

PARAMETERS

y Is the y (row) coordinate of the position of the character to be read.

x Is the x (column) coordinate of the position of the character to be read.

fmt Is a scanf () format string.

win Is a pointer to the window in which the character is to be read.

DESCRIPTION These functions are similar to scanf(3C). Their effect is as though mvwgetstr(3XCURSES) were called to get a multi-byte character string from the current or specified window at the current or specified cursor position, and then sscanf () were used to interpret and convert that string.

RETURN VALUES Upon successful completion, these functions return OK. Otherwise, they return ERR.

ERRORS No errors are defined.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO getnstr(3XCURSES), libcurses(3XCURSES), printw(3XCURSES), scanf(3C), wcstombs(3C), attributes(5), standards(5)

| | |
|----------------------|--|
| NAME | hline, mvhline, mvvline, mvwhline, mvwvline, vline, whline, wvline – use single-byte characters (and renditions) to draw lines |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int hline(chtype ch, int n); int mvhline(int y, int x, chtype ch, int n); int mvvline(int y, int x, chtype ch, int n); int mvwhline(WINDOW *win, int y, int x, chtype ch, int n); int mvwvline(WINDOW *win, int y, int x, chtype ch, int n); int vline(chtype ch, int n); int whline(WINDOW *win, chtype ch, int n); int wvline(WINDOW *win, chtype ch, int n);</pre> |
| DESCRIPTION | <p>The <code>hline()</code>, <code>vline()</code>, <code>whline()</code>, <code>wvline()</code> functions draw a horizontal or vertical line, in either the window <code>stdscr</code> or <code>win</code> starting at the current cursor position. The line is drawn using the character <code>ch</code> and is a maximum of <code>n</code> positions long, or as many as will fit into the window. If <code>ch</code> is 0 (zero), the default horizontal or vertical character is used.</p> <p>The <code>mvhline()</code>, <code>mvvline()</code>, <code>mvwhline()</code>, <code>mvwvline()</code> functions are similar to the previous group of functions but the line begins at cursor position specified by <code>x</code> and <code>y</code>.</p> <p>The functions with names ending with <code>hline()</code> draw horizontal lines proceeding towards the last column of the same line. The functions with names ending with <code>vline()</code> draw vertical lines proceeding towards the last column of the same line.</p> <p>These functions do not change the position of the cursor.</p> |
| PARAMETERS | <p><code>ch</code> Is the character used to draw the line.</p> <p><code>n</code> Is the maximum number of characters in the line.</p> <p><code>y</code> Is the y (row) coordinate for the start of the line.</p> <p><code>x</code> Is the x (column) coordinate for the start of the line.</p> <p><code>win</code> Is a pointer to a window.</p> |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. |
| ERRORS | None |

mvvline(3XCURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `border(3XCURSES)`, `border_set(3XCURSES)`, `hline_set(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|----------------------|---|
| NAME | hline_set, mvhline_set, mvvline_set, mvwhline_set, mvvwline_set, vline_set, whline_set, wvline_set – use complex characters (and renditions) to draw lines |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int hline_set(const cchar_t *ch, int n); int mvhline_set(int y, int x, const cchar_t *wch, int n); int mvvline_set(int y, int x, const cchar_t *wch, int n); int mvwhline_set(WINDOW *win, int y, int x, const cchar_t *wch, int n); int mvvwline_set(WINDOW *win, int y, int x, const cchar_t *wch, int n); int vline_set(const cchar_t *wch, int n); int whline_set(WINDOW *win, const cchar_t *wch, int n); int wvline_set(WINDOW *win, const cchar_t *wch, int n);</pre> |
| DESCRIPTION | <p>The <code>hline_set()</code>, <code>vline_set()</code>, <code>whline_set()</code>, <code>wvline_set()</code> functions draw a line, in either the window <code>stdscr</code> or <code>win</code> starting at the current cursor position. The line is drawn using the character <code>wch</code> and is a maximum of <code>n</code> positions long, or as many as will fit into the window. If <code>wch</code> is a null pointer, the default horizontal or vertical character is used.</p> <p>The <code>mvhline_set()</code>, <code>mvvline_set()</code>, <code>mvwhline_set()</code>, <code>mvvwline_set()</code> functions are similar to the previous group of functions but the line begins at cursor position specified by <code>x</code> and <code>y</code>.</p> <p>The functions with names ending with <code>hline_set()</code> draw horizontal lines proceeding towards the last column of the same line. The functions with names ending with <code>vline_set()</code> draw vertical lines proceeding towards the last column of the same line.</p> <p>These functions do not change the position of the cursor.</p> |
| PARAMETERS | <p><code>wch</code> Is the complex character used to draw the line.</p> <p><code>n</code> Is the maximum number of characters in the line.</p> <p><code>y</code> Is the y (row) coordinate for the start of the line.</p> <p><code>x</code> Is the x (column) coordinate for the start of the line.</p> <p><code>win</code> Is a pointer to a window.</p> |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. |

mvvline_set(3XCURSES)

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `border(3XCURSES)`, `border_set(3XCURSES)`, `hline(3XCURSES)`,
`libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|----------------------|--|
| NAME | curs_addch, addch, waddch, mvaddch, mvwaddch, echochar, wechochar – add a character (with attributes) to a curses window and advance cursor |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int addch(chtype ch); int waddch(WINDOW *win, chtype ch); int mvaddch(int y, int x, chtype ch); int mvwaddch(WINDOW *win, int y, int x, chtype ch); int echochar(chtype ch); int wechochar(WINDOW *win, chtype ch);</pre> |
| DESCRIPTION | <p>With the <code>addch()</code>, <code>waddch()</code>, <code>mvaddch()</code>, and <code>mvwaddch()</code> routines, the character <code>ch</code> is put into the window at the current cursor position of the window and the position of the window cursor is advanced. Its function is similar to that of <code>putchar()</code>. At the right margin, an automatic newline is performed. At the bottom of the scrolling region, if <code>scrollok()</code> is enabled, the scrolling region is scrolled up one line.</p> <p>If <code>ch</code> is a tab, newline, or backspace, the cursor is moved appropriately within the window. A newline also does a <code>clrtoeol()</code> before moving. Tabs are considered to be at every eighth column. If <code>ch</code> is another control character, it is drawn in the <code>^X</code> notation. Calling <code>winch()</code> after adding a control character does not return the control character, but instead returns the representation of the control character. See <code>curs_inch(3CURSES)</code>.</p> <p>Video attributes can be combined with a character by OR-ing them into the parameter. This results in these attributes also being set. (The intent here is that text, including attributes, can be copied from one place to another using <code>inch()</code> and <code>addch()</code>.) (see <code>standout()</code>, predefined video attribute constants, on the <code>curs_attr(3CURSES)</code> page).</p> <p>The <code>echochar()</code> and <code>wechochar()</code> routines are functionally equivalent to a call to <code>addch()</code> followed by a call to <code>refresh()</code>, or a call to <code>waddch</code> followed by a call to <code>wrefresh()</code>. The knowledge that only a single character is being output is taken into consideration and, for non-control characters, a considerable performance gain might be seen by using these routines instead of their equivalents.</p> |
| Line Graphics | <p>The following variables may be used to add line drawing characters to the screen with routines of the <code>addch()</code> family. When variables are defined for the terminal, the <code>A_ALTCHARSET</code> bit is turned on (see <code>curs_attr(3CURSES)</code>). Otherwise, the default character listed below is stored in the variable. The names chosen are consistent with the VT100 nomenclature.</p> |

mvwaddch(3CURSES)

| Name | Default | Glyph Description |
|--------------|---------|-------------------------|
| ACS_ULCORNER | + | upper left-hand corner |
| ACS_LLCORNER | + | lower left-hand corner |
| ACS_URCORNER | + | upper right-hand corner |
| ACS_LRCORNER | + | lower right-hand corner |
| ACS_RTEE | + | right tee |
| ACS_LTEE | + | left tee |
| ACS_BTEE | + | bottom tee |
| ACS_TTEE | + | top tee |
| ACS_HLINE | - | horizontal line |
| ACS_VLINE | | vertical line |
| ACS_PLUS | + | plus |
| ACS_S1 | - | scan line 1 |
| ACS_S9 | - | scan line 9 |
| ACS_DIAMOND | + | diamond |
| ACS_CKBOARD | : | checker board (stipple) |
| ACS_DEGREE | ' | degree symbol |
| ACS_PLMINUS | # | plus/minus |
| ACS_BULLET | o | bullet |
| ACS_LARROW | < | arrow pointing left |
| ACS_RARROW | > | arrow pointing right |
| ACS_DARROW | v | arrow pointing down |
| ACS_UARROW | ^ | arrow pointing up |
| ACS_BOARD | # | board of squares |
| ACS_LANTERN | # | lantern symbol |
| ACS_BLOCK | # | solid square block |

RETURN VALUES

All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

mvwaddch(3CURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_attr(3CURSES)`, `curs_clear(3CURSES)`, `curs_inch(3CURSES)`,
 `curs_outopts(3CURSES)`, `curs_refresh(3CURSES)`, `curses(3CURSES)`,
 `putc(3C)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and
`<unctrl.h>`.

Note that `addch()`, `mvaddch()`, `mvwaddch()`, and `echochar()` may be macros.

mvwaddch(3XCURSES)

| | | | | | | | | | | | |
|----------------------|---|---------------|---|----------|--|----------|--|----------|---|------------|---|
| NAME | addch, mvaddch, mvwaddch, waddch – add a character (with rendition) to a window | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addch(const chtype ch); int mvaddch(int y, int x, const chtype ch); int mvwaddch(WINDOW *win, int y, int x, const chtype ch); int waddch(WINDOW *win, const chtype ch);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The addch() function writes a character to the stdscr window at the current cursor position. The mvaddch() and mvwaddch() functions write the character to the position indicated by the x (column) and y (row) parameters. The mvaddch() function writes the character to the stdscr window, while mvwaddch() writes the character to the window specified by win. The waddch() function is identical to addch(), but writes the character to the window specified by win.</p> <p>These functions advance the cursor after writing the character. Characters that do not fit on the end of the current line are wrapped to the beginning of the next line unless the current line is the last line of the window and scrolling is disabled. In that situation, characters which extend beyond the end of the line are discarded.</p> <p>When ch is a backspace, carriage return, newline, or tab, X/Open Curses moves the cursor appropriately. Each tab character moves the cursor to the next tab stop. By default, tab stops occur every eight columns. When ch is a control character other than backspace, carriage return, newline, or tab, it is written using ^x notation, where x is a printable character. When X/Open Curses writes ch to the last character position on a line, it automatically generates a newline. When ch is written to the last character position of a scrolling region and scrolllok() is enabled, X/Open Curses scrolls the scrolling region up one line (see clearok(3XCURSES)).</p> | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>wchstr</i></td> <td>Is a pointer to the <code>cchar_t</code> string to be copied to the window.</td> </tr> <tr> <td><i>n</i></td> <td>Is the maximum number of characters to be copied from <i>wchstr</i>. If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line.</td> </tr> <tr> <td><i>y</i></td> <td>Is the y (row) coordinate of the starting position of <i>wchstr</i> in the window.</td> </tr> <tr> <td><i>x</i></td> <td>Is the x (column) coordinate of the starting position of <i>wchstr</i> in the window.</td> </tr> <tr> <td><i>win</i></td> <td>Is a pointer to the window to which the string is to be copied.</td> </tr> </table> | <i>wchstr</i> | Is a pointer to the <code>cchar_t</code> string to be copied to the window. | <i>n</i> | Is the maximum number of characters to be copied from <i>wchstr</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | <i>y</i> | Is the y (row) coordinate of the starting position of <i>wchstr</i> in the window. | <i>x</i> | Is the x (column) coordinate of the starting position of <i>wchstr</i> in the window. | <i>win</i> | Is a pointer to the window to which the string is to be copied. |
| <i>wchstr</i> | Is a pointer to the <code>cchar_t</code> string to be copied to the window. | | | | | | | | | | |
| <i>n</i> | Is the maximum number of characters to be copied from <i>wchstr</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | | | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of the starting position of <i>wchstr</i> in the window. | | | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of the starting position of <i>wchstr</i> in the window. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window to which the string is to be copied. | | | | | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | | | | | |

mvwaddch(3XCURSES)

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attroff(3XCURSES)`, `bkgdset(3XCURSES)`, `douupdate(3XCURSES)`, `inch(3XCURSES)`, `insch(3XCURSES)`, `libcurses(3XCURSES)`, `nl(3XCURSES)`, `printw(3XCURSES)`, `scrollok(3XCURSES)`, `scl(3XCURSES)`, `terminfo(4)`, `attributes(5)`, `standards(5)`

mvwaddchnstr(3CURSES)

| NAME | curs_addchstr, addchstr, addchnstr, waddchstr, waddchnstr, mvaddchstr, mvaddchnstr, mvwaddchstr, mvwaddchnstr – add string of characters and attributes to a curses window | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int addchstr(chtype *chstr); int addchnstr(chtype *chstr, int n); int waddchstr(WINDOW *win, chtype *chstr); int waddchnstr(WINDOW *win, chtype *chstr, int n); int mvaddchstr(int y, int x, chtype *chstr); int mvaddchnstr(int y, int x, chtype *chstr, int n); int mvwaddchstr(WINDOW *win, int y, int x, chtype *chstr); int mvwaddchnstr(WINDOW *win, int y, int x, chtype *chstr, int n);</pre> | | | | |
| DESCRIPTION | <p>All of these routines copy <i>chstr</i> directly into the window image structure starting at the current cursor position. The four routines with <i>n</i> as the last argument copy at most <i>n</i> elements, but no more than will fit on the line. If <i>n</i> = -1 then the whole string is copied, to the maximum number that fit on the line.</p> <p>The position of the window cursor is not advanced. These routines works faster than <code>waddnstr()</code> (see <code>curs_addstr(3CURSES)</code>) because they merely copy <i>chstr</i> into the window image structure. On the other hand, care must be taken when using these functions because they do not perform any kind of checking (such as for the newline character), they do not advance the current cursor position, and they truncate the string, rather than wrapping it around to the next line.</p> | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_addstr(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that all routines except <code>waddchnstr()</code> and <code>waddchstr()</code> may be macros.</p> | | | | |

| | |
|----------------------|---|
| NAME | addchstr, addchnstr, mvaddchstr, mvaddchnstr, mvwaddchnstr, mvwaddchstr, waddchstr, waddchnstr – copy a character string (with renditions) to a window |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addchstr(const chtype *chstr); int addchnstr(const chtype *chstr, int n); int mvaddchnstr(int y, int x, const chtype *chstr, int n); int mvaddchstr(int y, int x, const chtype *chstr); int mvwaddchnstr(WINDOW *win, int y, int x, const chtype *chstr, int n); int mvwaddchstr(WINDOW *win, int y, int x, const chtype *chstr); int waddchstr(WINDOW *win, const chtype *chstr); int waddchnstr(WINDOW *win, const chtype *chstr, int n);</pre> |
| DESCRIPTION | <p>The <code>addchstr()</code> function copies the <code>chtype</code> character string to the <code>stdscr</code> window at the current cursor position. The <code>mvaddchstr()</code> and <code>mvwaddchstr()</code> functions copy the character string to the starting position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <code>win</code>). The <code>waddchstr()</code> is identical to <code>addchstr()</code>, but writes to the window specified by <code>win</code>.</p> <p>The <code>addchnstr()</code>, <code>waddchnstr()</code>, <code>mvaddchnstr()</code>, and <code>mvwaddchnstr()</code> functions write <code>n</code> characters to the window, or as many as will fit on the line. If <code>n</code> is less than 0, the entire string is written, or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <code>x</code> and <code>y</code> parameters.</p> <p>These functions differ from the <code>addstr(3XCURSES)</code> set of functions in two important respects. First, these functions do <i>not</i> advance the cursor after writing the string to the window. Second, the current window rendition is not combined with the character; only the attributes that are already part of the <code>chtype</code> character are used.</p> |
| PARAMETERS | <p><code>chstr</code> Is a pointer to the <code>chtype</code> string to be copied to the window.</p> <p><code>n</code> Is the maximum number of characters to be copied from <code>chstr</code>. If <code>n</code> is less than 0, the entire string is written or as much of it as fits on the line.</p> <p><code>y</code> Is the <code>y</code> (row) coordinate of the starting position of <code>chstr</code> in the window.</p> <p><code>x</code> Is the <code>x</code> (column) coordinate of the starting position of <code>chstr</code> in the window.</p> <p><code>win</code> Is a pointer to the window to which the string is to be copied.</p> |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. |

mvwaddchnstr(3XCURSES)

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addch(3XCURSES)`, `addnstr(3XCURSES)`, `attroff(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_addchstr, addchstr, addchnstr, waddchstr, waddchnstr, mvaddchstr, mvaddchnstr, mvwaddchstr, mvwaddchnstr – add string of characters and attributes to a curses window | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int addchstr(chtype *chstr); int addchnstr(chtype *chstr, int n); int waddchstr(WINDOW *win, chtype *chstr); int waddchnstr(WINDOW *win, chtype *chstr, int n); int mvaddchstr(int y, int x, chtype *chstr); int mvaddchnstr(int y, int x, chtype *chstr, int n); int mvwaddchstr(WINDOW *win, int y, int x, chtype *chstr); int mvwaddchnstr(WINDOW *win, int y, int x, chtype *chstr, int n);</pre> | | | | |
| DESCRIPTION | <p>All of these routines copy <i>chstr</i> directly into the window image structure starting at the current cursor position. The four routines with <i>n</i> as the last argument copy at most <i>n</i> elements, but no more than will fit on the line. If <i>n</i> = -1 then the whole string is copied, to the maximum number that fit on the line.</p> <p>The position of the window cursor is not advanced. These routines works faster than <code>waddnstr()</code> (see <code>curs_addstr(3CURSES)</code>) because they merely copy <i>chstr</i> into the window image structure. On the other hand, care must be taken when using these functions because they do not perform any kind of checking (such as for the newline character), they do not advance the current cursor position, and they truncate the string, rather than wrapping it around to the next line.</p> | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_addstr(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that all routines except <code>waddchnstr()</code> and <code>waddchstr()</code> may be macros.</p> | | | | |

mvwaddchstr(3XCURSES)

| | | | | | | | | | | | |
|----------------------|---|--------------------|--|----------------|---|----------------|--|----------------|---|------------------|---|
| NAME | addchstr, addchnstr, mvaddchstr, mvaddchnstr, mvwaddchnstr, mvwaddchstr, waddchstr, waddchnstr – copy a character string (with renditions) to a window | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addchstr(const chtype *chstr); int addchnstr(const chtype *chstr, int n); int mvaddchnstr(int y, int x, const chtype *chstr, int n); int mvaddchstr(int y, int x, const chtype *chstr); int mvwaddchnstr(WINDOW *win, int y, int x, const chtype *chstr, int n); int mvwaddchstr(WINDOW *win, int y, int x, const chtype *chstr); int waddchstr(WINDOW *win, const chtype *chstr); int waddchnstr(WINDOW *win, const chtype *chstr, int n);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The <code>addchstr()</code> function copies the <code>chtype</code> character string to the <code>stdscr</code> window at the current cursor position. The <code>mvaddchstr()</code> and <code>mvwaddchstr()</code> functions copy the character string to the starting position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <code>win</code>). The <code>waddchstr()</code> is identical to <code>addchstr()</code>, but writes to the window specified by <code>win</code>.</p> <p>The <code>addchnstr()</code>, <code>waddchnstr()</code>, <code>mvaddchnstr()</code>, and <code>mvwaddchnstr()</code> functions write <code>n</code> characters to the window, or as many as will fit on the line. If <code>n</code> is less than 0, the entire string is written, or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <code>x</code> and <code>y</code> parameters.</p> <p>These functions differ from the <code>addstr(3XCURSES)</code> set of functions in two important respects. First, these functions do <i>not</i> advance the cursor after writing the string to the window. Second, the current window rendition is not combined with the character; only the attributes that are already part of the <code>chtype</code> character are used.</p> | | | | | | | | | | |
| PARAMETERS | <table><tr><td><code>chstr</code></td><td>Is a pointer to the <code>chtype</code> string to be copied to the window.</td></tr><tr><td><code>n</code></td><td>Is the maximum number of characters to be copied from <code>chstr</code>. If <code>n</code> is less than 0, the entire string is written or as much of it as fits on the line.</td></tr><tr><td><code>y</code></td><td>Is the <code>y</code> (row) coordinate of the starting position of <code>chstr</code> in the window.</td></tr><tr><td><code>x</code></td><td>Is the <code>x</code> (column) coordinate of the starting position of <code>chstr</code> in the window.</td></tr><tr><td><code>win</code></td><td>Is a pointer to the window to which the string is to be copied.</td></tr></table> | <code>chstr</code> | Is a pointer to the <code>chtype</code> string to be copied to the window. | <code>n</code> | Is the maximum number of characters to be copied from <code>chstr</code> . If <code>n</code> is less than 0, the entire string is written or as much of it as fits on the line. | <code>y</code> | Is the <code>y</code> (row) coordinate of the starting position of <code>chstr</code> in the window. | <code>x</code> | Is the <code>x</code> (column) coordinate of the starting position of <code>chstr</code> in the window. | <code>win</code> | Is a pointer to the window to which the string is to be copied. |
| <code>chstr</code> | Is a pointer to the <code>chtype</code> string to be copied to the window. | | | | | | | | | | |
| <code>n</code> | Is the maximum number of characters to be copied from <code>chstr</code> . If <code>n</code> is less than 0, the entire string is written or as much of it as fits on the line. | | | | | | | | | | |
| <code>y</code> | Is the <code>y</code> (row) coordinate of the starting position of <code>chstr</code> in the window. | | | | | | | | | | |
| <code>x</code> | Is the <code>x</code> (column) coordinate of the starting position of <code>chstr</code> in the window. | | | | | | | | | | |
| <code>win</code> | Is a pointer to the window to which the string is to be copied. | | | | | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | | | | | |

mvwaddchstr(3XCURSES)

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addch(3XCURSES)`, `addnstr(3XCURSES)`, `attroff(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

mvwaddnstr(3CURSES)

NAME | curs_addstr, addstr, addnstr, waddstr, waddnstr, mvaddstr, mvaddnstr, mvwaddstr, mvwaddnstr – add a string of characters to a curses window and advance cursor

SYNOPSIS |

```
cc [ flag ... ] file ... -lcurses [ library ... ]
#include <curses.h>

int addstr(char *str);
int addnstr(char *str, int n);
int waddstr(WINDOW *win, char *str);
int waddnstr(WINDOW *win, char *str, int n);
int mvaddstr(int y, int x, char *str);
int mvaddnstr(int y, int x, char *str, int n);
int mvwaddstr(WINDOW *win, int y, int x, char *str);
int mvwaddnstr(WINDOW *win, int y, int x, char *str, int n);
```

DESCRIPTION | All of these routines write all the characters of the null terminated character string *str* on the given window. It is similar to calling `waddch()` once for each character in the string. The four routines with *n* as the last argument write at most *n* characters. If *n* is negative, then the entire string will be added.

RETURN VALUES | All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | `curs_addch(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES | The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that all routines except `waddstr()` and `waddnstr()` may not be macros.

| | |
|--------------------|---|
| NAME | addnstr, addstr, mvaddnstr, mvaddstr, mvwaddnstr, mvwaddstr, waddnstr, waddstr – add a multi-byte character string (without rendition) to a window |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addnstr(const char *str, int n); int addstr(const char *str); int mvaddnstr(int y, int x, const char *str, int n); int mvaddstr(int y, int x, const char *str); int mvwaddnstr(WINDOW *win, int y, int x, const char *str, int n); int mvwaddstr(WINDOW *win, int y, int x, const char *str); int waddstr(WINDOW *win, const char *str); int waddnstr(WINDOW *win, const char *str, int n);</pre> |
| DESCRIPTION | <p>The <code>addstr()</code> function writes a null-terminated string of multi-byte characters to the <code>stdscr</code> window at the current cursor position. The <code>waddstr()</code> function performs an identical action, but writes the character to the window specified by <i>win</i>. The <code>mvaddstr()</code> and <code>mvwaddstr()</code> functions write the string to the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <i>win</i>).</p> <p>The <code>addnstr()</code>, <code>waddnstr()</code>, <code>mvaddnstr()</code>, and <code>mvwaddnstr()</code> functions are similar but write at most <i>n</i> characters to the window. If <i>n</i> is less than 0, the entire string is written.</p> <p>All of these functions advance the cursor after writing the string.</p> <p>These functions are functionally equivalent to calling the corresponding function from the <code>addch(3XCURSES)</code> set of functions once for each character in the string. Refer to the <code>curses(3XCURSES)</code> man page for a complete description of special character handling and of the interaction between the window rendition (or background character and rendition) and the character written.</p> <p>Note that these functions differ from the <code>addchstr()</code> set of functions in that the <code>addchstr(3XCURSES)</code> functions copy the string as is (without combining each character with the window rendition or the background character and rendition).</p> |
| PARAMETERS | <p><i>str</i> Is a pointer to the character string that is to be written to the window.</p> <p><i>n</i> Is the maximum number of characters to be copied from <i>str</i>. If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line.</p> <p><i>y</i> Is the <i>y</i> (row) coordinate of the starting position of <i>str</i> in the window.</p> |

mvwaddnstr(3XCURSES)

x Is the x (column) coordinate of the starting position of *str* in the window.

win Is a pointer to the window in which the string is to be written.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addch(3XCURSES)`, `addchstr(3XCURSES)`, `curses(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_addwstr, addwstr, addnwstr, waddwstr, waddnwstr, mvaddwstr, mvaddnwstr, mvwaddwstr, mvwaddnwstr – add a string of wchar_t characters to a curses window and advance cursor | | | | |
|---------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lcurses [library...] #include<curses.h> int addwstr(wchar_t *wstr); int addnwstr(wchar_t *wstr, int n); int waddwstr(WINDOW *win, wchar_t *wstr); int waddnwstr(WINDOW *win, wchar_t *wstr, int n); int mvaddwstr(int y, int x, wchar_t *wstr); int mvaddnwstr(int y, int x, wchar_t *wstr, int n); int mvwaddwstr(WINDOW *win, int y, int x, wchar_t *wstr); int mvwaddnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);</pre> | | | | |
| DESCRIPTION | All of these routines write all the characters of the null-terminated wchar_t character string wstr on the given window. The effect is similar to calling waddwch(3CURSES) once for each wchar_t character in the string. The four routines with n as the last argument write at most n wchar_t characters. If n is negative, then the entire string will be added. | | | | |
| RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), waddwch(3CURSES), attributes(5) | | | | |
| NOTES | <p>The header file <curses.h> automatically includes the header files <stdio.h>, <nctrl.h> and <widec.h>.</p> <p>Note that all of these routines except waddwstr() and waddnwstr() may be macros.</p> | | | | |

mvwaddnwstr(3XCURSES)

| | |
|--------------------|---|
| NAME | <code>addnwstr</code> , <code>addwstr</code> , <code>mvaddnwstr</code> , <code>mvaddwstr</code> , <code>mvwaddnwstr</code> , <code>mvwaddwstr</code> , <code>waddnwstr</code> , <code>waddwstr</code> – add a wide-character string to a window |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addnwstr(const wchar_t *wstr, int n); int addwstr(const wchar_t *wstr); int mvaddnwstr(int y, int x, const wchar_t *wstr, int n); int mvaddwstr(int y, int x, const wchar_t *wstr); int mvwaddnwstr(WINDOW*win, int y, int x, const wchar_t *wstr, int n); int mvwaddwstr(WINDOW*win, int y, int x, const wchar_t *wstr); int waddnwstr(WINDOW*win, const wchar_t *wstr, int n); int waddwstr(WINDOW*win, const wchar_t *wstr);</pre> |
| DESCRIPTION | <p>The <code>addwstr()</code> function writes a null-terminated wide-character string to the <code>stdscr</code> window at the current cursor position. The <code>waddwstr()</code> function performs an identical action, but writes the string to the window specified by <code>win</code>. The <code>mvaddwstr()</code> and <code>mvwaddwstr()</code> functions write the string to the position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <code>win</code>).</p> <p>The <code>addnwstr()</code>, <code>waddnwstr()</code>, <code>mvaddnwstr()</code>, and <code>mvwaddnwstr()</code> functions write at most <code>n</code> characters to the window. If <code>n</code> is less than 0, the entire string is written. The former two functions place the characters at the current cursor position; the latter two commands use the position specified by the <code>x</code> and <code>y</code> parameters.</p> <p>All of these functions advance the cursor after writing the string.</p> <p>These functions are functionally equivalent to building a <code>cchar_t</code> from the <code>wchar_t</code> and the window rendition (or background character and rendition) and calling the <code>wadd_wch(3XCURSES)</code> function once for each <code>wchar_t</code> in the string. Refer to the <code>curses(3XCURSES)</code> man page for a complete description of special character handling and of the interaction between the window rendition (or background character and rendition) and the character written.</p> <p>Note that these functions differ from the <code>add_wchnstr(3XCURSES)</code> set of functions in that the latter copy the string as is (without combining each character with the foreground and background attributes of the window).</p> |
| PARAMETERS | <p><code>wstr</code> Is a pointer to the wide-character string that is to be written to the window.</p> |

mvwaddnwstr(3XCURSES)

n Is the maximum number of characters to be copied from *wstr*. If *n* is less than 0, the entire string is written or as much of it as fits on the line.

y Is the y (row) coordinate of the starting position of *wstr* in the window.

x Is the x (column) coordinate of the starting position of *wstr* in the window.

win Is a pointer to the window in which the string is to be written.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `add_wch(3XCURSES)`, `add_wchnstr(3XCURSES)`, `curses(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

mvwaddstr(3CURSES)

| NAME | <code> curs_ addstr, addstr, addnstr, waddstr, waddnstr, mvaddstr, mvaddnstr, mvwaddstr, mvwaddnstr </code> – add a string of characters to a curses window and advance cursor | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int addstr(char *str); int addnstr(char *str, int n); int waddstr(WINDOW *win, char *str); int waddnstr(WINDOW *win, char *str, int n); int mvaddstr(int y, int x, char *str); int mvaddnstr(int y, int x, char *str, int n); int mvwaddstr(WINDOW *win, int y, int x, char *str); int mvwaddnstr(WINDOW *win, int y, int x, char *str, int n);</pre> | | | | |
| DESCRIPTION | All of these routines write all the characters of the null terminated character string <i>str</i> on the given window. It is similar to calling <code>waddch()</code> once for each character in the string. The four routines with <i>n</i> as the last argument write at most <i>n</i> characters. If <i>n</i> is negative, then the entire string will be added. | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code> curs_ addch(3CURSES), curses(3CURSES), attributes(5) </code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . Note that all routines except <code>waddstr()</code> and <code>waddnstr()</code> may not be macros. | | | | |

| | |
|--------------------|---|
| NAME | addnstr, addstr, mvaddnstr, mvaddstr, mvwaddnstr, mvwaddstr, waddnstr, waddstr – add a multi-byte character string (without rendition) to a window |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addnstr(const char *str, int n); int addstr(const char *str); int mvaddnstr(int y, int x, const char *str, int n); int mvaddstr(int y, int x, const char *str); int mvwaddnstr(WINDOW *win, int y, int x, const char *str, int n); int mvwaddstr(WINDOW *win, int y, int x, const char *str); int waddstr(WINDOW *win, const char *str); int waddnstr(WINDOW *win, const char *str, int n);</pre> |
| DESCRIPTION | <p>The <code>addstr()</code> function writes a null-terminated string of multi-byte characters to the <code>stdscr</code> window at the current cursor position. The <code>waddstr()</code> function performs an identical action, but writes the character to the window specified by <i>win</i>. The <code>mvaddstr()</code> and <code>mvwaddstr()</code> functions write the string to the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <i>win</i>).</p> <p>The <code>addnstr()</code>, <code>waddnstr()</code>, <code>mvaddnstr()</code>, and <code>mvwaddnstr()</code> functions are similar but write at most <i>n</i> characters to the window. If <i>n</i> is less than 0, the entire string is written.</p> <p>All of these functions advance the cursor after writing the string.</p> <p>These functions are functionally equivalent to calling the corresponding function from the <code>addch(3XCURSES)</code> set of functions once for each character in the string. Refer to the <code>curses(3XCURSES)</code> man page for a complete description of special character handling and of the interaction between the window rendition (or background character and rendition) and the character written.</p> <p>Note that these functions differ from the <code>addchstr()</code> set of functions in that the <code>addchstr(3XCURSES)</code> functions copy the string as is (without combining each character with the window rendition or the background character and rendition).</p> |
| PARAMETERS | <p><i>str</i> Is a pointer to the character string that is to be written to the window.</p> <p><i>n</i> Is the maximum number of characters to be copied from <i>str</i>. If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line.</p> <p><i>y</i> Is the <i>y</i> (row) coordinate of the starting position of <i>str</i> in the window.</p> |

`mvwaddstr(3XCURSES)`

x Is the x (column) coordinate of the starting position of *str* in the window.

win Is a pointer to the window in which the string is to be written.

RETURN VALUES On success, these functions return `OK`. Otherwise, they return `ERR`.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addch(3XCURSES)`, `addchstr(3XCURSES)`, `curses(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|----------------------|---|
| NAME | curs_addwch, addwch, waddwch, mvaddwch, mvwaddwch, echowchar, wechowchar – add a wchar_t character (with attributes) to a curses window and advance cursor |
| SYNOPSIS | <pre>cc [flag...] file... -lcurses [library...] #include<curses.h> int addwch(chtype wch); int waddwch(WINDOW *win, chtype wch); int mvaddwch(int y, int x, chtype wch); int mvwaddwch(WINDOW *win, int y, int x, chtype wch); int echowchar(chtype wch); int wechowchar(WINDOW *win, chtype wch);</pre> |
| DESCRIPTION | <p>The addwch(), waddwch(), mvaddwch(), and mvwaddwch() routines put the character <i>wch</i>, holding a wchar_t character, into the window at the current cursor position of the window and advance the position of the window cursor. Their function is similar to that of putwchar(3C) in the C multibyte library. At the right margin, an automatic newline is performed. At the bottom of the scrolling region, if scrolllok is enabled, the scrolling region is scrolled up one line.</p> <p>If <i>wch</i> is a tab, newline, or backspace, the cursor is moved appropriately within the window. A newline also does a clrtoeol(3CURSES) before moving. Tabs are considered to be at every eighth column. If <i>wch</i> is another control character, it is drawn in the ^X notation. Calling winwch(3CURSES) after adding a control character does not return the control character, but instead returns the representation of the control character.</p> <p>Video attributes can be combined with a wchar_t character by OR-ing them into the parameter. This results in these attributes also being set. (The intent here is that text, including attributes, can be copied from one place to another using inwch() and addwch().) See standout(3CURSES), predefined video attribute constants.</p> <p>The echowchar() and wechowchar() routines are functionally equivalent to a call to addwch() followed by a call to refresh(3CURSES), or a call to waddwch() followed by a call to wrefresh(3CURSES). The knowledge that only a single character is being output is taken into consideration and, for non-control characters, a considerable performance gain might be seen by using these routines instead of their equivalents.</p> |
| Line Graphics | <p>The following variables may be used to add line drawing characters to the screen with routines of the addwch() family. When variables are defined for the terminal, the A_ALTCHARSET bit is turned on. (See curs_attr(3CURSES)). Otherwise, the default character listed below is stored in the variable. The names chosen are consistent with the VT100 nomenclature.</p> |

mvwaddwch(3CURSES)

| Name | Default | Glyph Description |
|--------------|---------|-------------------------|
| ACS_ULCORNER | + | upper left-hand corner |
| ACS_LLCORNER | + | lower left-hand corner |
| ACS_URCORNER | + | upper right-hand corner |
| ACS_LRCORNER | + | lower right-hand corner |
| ACS_RTEE | + | right tee |
| ACS_LTEE | + | left tee |
| ACS_BTEE | + | bottom tee |
| ACS_TTEE | + | top tee |
| ACS_HLINE | - | horizontal line |
| ACS_VLINE | | vertical line |
| ACS_PLUS | + | plus |
| ACS_S1 | - | scan line 1 |
| ACS_S9 | - | scan line 9 |
| ACS_DIAMOND | + | diamond |
| ACS_CKBOARD | : | checker board (stipple) |
| ACS_DEGREE | ' | degree symbol |
| ACS_PLMINUS | # | plus/minus |
| ACS_BULLET | o | bullet |
| ACS_LARROW | < | arrow pointing left |
| ACS_RARROW | > | arrow pointing right |
| ACS_DARROW | v | arrow pointing down |
| ACS_UARROW | ^ | arrow pointing up |
| ACS_BOARD | # | board of squares |
| ACS_LANTERN | # | lantern symbol |
| ACS_BLOCK | # | solid square block |

RETURN VALUE

All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion, unless otherwise noted in the preceding routine descriptions.

mvwaddwch(3CURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `putwchar(3C)`, `clrtoeol(3CURSES)`, `curses(3CURSES)`, `curs_attr(3CURSES)`, `curs_inwch(3CURSES)`, `curs_outopts(3CURSES)`, `refresh(3CURSES)`, `standout(3CURSES)`, `winwch(3CURSES)`, `wrefresh(3CURSES)`, `attributes(5)`

NOTES The header file `<curses.h>` automatically includes the header files `<stdio.h>`, `<unctrl.h>` and `<widec.h>`.

Note that `addwch()`, `mvaddwch()`, `mvwaddwch()`, and `echowchar()` may be macros.

None of these routines can use the color attribute in `chtype`.

mvwadd_wch(3XCURSES)

| | |
|--------------------|--|
| NAME | add_wch, mvadd_wch, mvwadd_wch, wadd_wch – add a complex character (with rendition) to a window |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int add_wch(const cchar_t *wch); int wadd_wch(WINDOW *win, const cchar_t *wch); int mvadd_wch(int y, int x, const cchar_t *wch); int mvwadd_wch(WINDOW *win, int y, int x, const cchar_t *wch);</pre> |
| DESCRIPTION | <p>The <code>add_wch()</code> function writes a complex character to the <code>stdscr</code> window at the current cursor position. The <code>mvadd_wch()</code> and <code>mvwadd_wch()</code> functions write the character to the position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters. The <code>mvadd_wch()</code> function writes the character to the <code>stdscr</code> window, while <code>mvwadd_wch()</code> writes the character to the window specified by <code>win</code>. The <code>wadd_wch()</code> function is identical to <code>add_wch()</code>, but writes the character to the window specified by <code>win</code>. These functions advance the cursor after writing the character.</p> <p>If <code>wch</code> is a spacing complex character, X/Open Curses replaces any previous character at the specified location with <code>wch</code> (and its rendition). If <code>wch</code> is a non-spacing complex character, X/Open Curses preserves all existing characters at the specified location and adds the non-spacing characters of <code>wch</code> to the spacing complex character. It ignores the rendition associated with <code>wch</code>.</p> <p>Characters that do not fit on the end of the current line are wrapped to the beginning of the next line unless the current line is the last line of the window and scrolling is disabled. In that situation, X/Open Curses discards characters which extend beyond the end of the line.</p> <p>When <code>wch</code> is a backspace, carriage return, newline, or tab, X/Open Curses moves the cursor appropriately as described in the <code>curses(3XCURSES)</code> man page. Each tab character moves the cursor to the next tab stop. By default, tab stops occur every eight columns. When <code>wch</code> is a control character other than a backspace, carriage return, newline, or tab, it is written using <code>^x</code> notation, where <code>x</code> is a printable character. When X/Open Curses writes <code>wch</code> to the last character position on a line, it automatically generates a newline. When <code>wch</code> is written to the last character position of a scrolling region and <code>scrollok()</code> is enabled, X/Open Curses scrolls the scrolling region up one line (see <code>clearok(3XCURSES)</code>).</p> |
| PARAMETERS | <p><code>wch</code> Is the character/attribute pair (rendition) to be written to the window.</p> <p><code>win</code> Is a pointer to the window in which the character is to be written.</p> <p><code>y</code> Is the <code>y</code> (row) coordinate of the character's position in the window.</p> |

mvwadd_wch(3XCURSES)

x Is the *x* (column) coordinate of the character's position in the window.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attr_off(3XCURSES)`, `bkgrndset(3XCURSES)`, `curses(3XCURSES)`, `doupdate(3XCURSES)`, `in_wch(3XCURSES)`, `ins_wch(3XCURSES)`, `libcurses(3XCURSES)`, `n1(3XCURSES)`, `printw(3XCURSES)`, `scrollok(3XCURSES)`, `scl1(3XCURSES)`, `setscrreg(3XCURSES)`, `terminfo(4)`, `attributes(5)`, `standards(5)`

mvwaddwchnstr(3CURSES)

| NAME | curs_addwchstr, addwchstr, addwchnstr, waddwchstr, waddwchnstr, mvaddwchstr, mvaddwchnstr, mvwaddwchstr, mvwaddwchnstr – add string of wchar_t characters (and attributes) to a curses window | | | | |
|---------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lcurses [library...] #include<curses.h> int addwchstr (chtype *wchstr); int addwchnstr (chtype *wchstr, int n); int waddwchstr (WINDOW *win, chtype *wchstr); int waddwchnstr (WINDOW *win, chtype *wchstr, int n); int mvaddwchstr (int y, int x, chtype *wchstr); int mvaddwchnstr (int y, int x, chtype *wchstr, int n); int mvwaddwchstr (WINDOW *win, int y, int x, chtype *wchstr); int mvwaddwchnstr (WINDOW *win, int y, int x, chtype *wchstr, int n);</pre> | | | | |
| DESCRIPTION | <p>All of these routines copy <i>wchstr</i>, which points to a string of wchar_t characters, directly into the window image structure starting at the current cursor position. The four routines with <i>n</i> as the last argument copy at most <i>n</i> elements, but no more than will fit on the line. If <i>n</i>==-1 then the whole string is copied, to the maximum number that fit on the line.</p> <p>The position of the window cursor is not advanced. These routines work faster than waddnwstr(3CURSES) because they merely copy <i>wchstr</i> into the window image structure. On the other hand, care must be taken when using these functions because they don't perform any kind of checking (such as for the newline character), they do not advance the current cursor position, and they truncate the string, rather than wrapping it around to the new line.</p> | | | | |
| RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion, unless otherwise noted in the preceding routine descriptions. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), waddnwstr(3CURSES), attributes(5) | | | | |
| NOTES | The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <widec.h>. | | | | |

`mvwaddwchnstr(3CURSES)`

Note that all routines except `waddwchnstr()` may be macros.

None of these routines can use the color attribute in `chtype`.

mvwadd_wchnstr(3XCURSES)

| | | | | | | | |
|--------------------|--|---------------|--|----------|--|----------|--|
| NAME | add_wchnstr, add_wchstr, mvadd_wchnstr, mvadd_wchstr, mvwadd_wchnstr, mvwadd_wchstr, wadd_wchnstr, wadd_wchstr – copy a string of complex characters (with renditions) to a window | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int add_wchnstr(const cchar_t *wchstr, int n); int add_wchstr(const cchar_t *wchstr); int mvadd_wchnstr(int y, int x, const cchar_t *wchstr, int n); int mvadd_wchstr(int y, int x, const cchar_t *wchstr); int mvwadd_wchnstr(WINDOW *win, int y, int x, const cchar_t *wchstr, int n); int mvwadd_wchstr(WINDOW *win, int y, int x, const cchar_t *wchstr); int wadd_wchstr(WINDOW *win, const cchar_t *wchstr); int wadd_wchnstr(WINDOW *win, const cchar_t *wchstr, int n);</pre> | | | | | | |
| DESCRIPTION | <p>The add_wchstr() function copies the string of cchar_t characters to the stdscr window at the current cursor position. The mvadd_wchstr() and mvwadd_wchstr() functions copy the string to the starting position indicated by the x (column) and y (row) parameters (the former to the stdscr window; the latter to window win). The wadd_wchstr() is identical to add_wchstr(), but writes to the window specified by win.</p> <p>The add_wchnstr(), wadd_wchnstr(), mvadd_wchnstr(), and mvwadd_wchnstr() functions write n characters to the window, or as many as will fit on the line. If n is less than 0, the entire string is written, or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the x and y parameters.</p> <p>These functions differ from the addwstr(3XCURSES) set of functions in two important respects. First, these functions do <i>not</i> advance the cursor after writing the string to the window. Second, the current window rendition (that is, the combination of attributes and color pair) is not combined with the character; only those attributes that are already part of the cchar_t character are used.</p> | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>wchstr</i></td> <td>Is a pointer to the cchar_t string to be copied to the window.</td> </tr> <tr> <td><i>n</i></td> <td>Is the maximum number of characters to be copied from <i>wchstr</i>. If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line.</td> </tr> <tr> <td><i>y</i></td> <td>Is the y (row) coordinate of the starting position of <i>wchstr</i> in the window.</td> </tr> </table> | <i>wchstr</i> | Is a pointer to the cchar_t string to be copied to the window. | <i>n</i> | Is the maximum number of characters to be copied from <i>wchstr</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | <i>y</i> | Is the y (row) coordinate of the starting position of <i>wchstr</i> in the window. |
| <i>wchstr</i> | Is a pointer to the cchar_t string to be copied to the window. | | | | | | |
| <i>n</i> | Is the maximum number of characters to be copied from <i>wchstr</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | | | | | | |
| <i>y</i> | Is the y (row) coordinate of the starting position of <i>wchstr</i> in the window. | | | | | | |

mvwadd_wchnstr(3XCURSES)

x Is the *x* (column) coordinate of the starting position of *wchstr* in the window.

win Is a pointer to the window to which the string is to be copied.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addnwstr(3XCURSES)`, `add_wch(3XCURSES)`, `attr_off(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

mvwaddwchstr(3CURSES)

| NAME | <p> <code> curs_addwchstr, addwchstr, addwchnstr, waddwchstr, waddwchnstr, mvaddwchstr, mvaddwchnstr, mvwaddwchstr, mvwaddwchnstr</code> – add string of <code>wchar_t</code> characters (and attributes) to a curses window </p> | | | | |
|---------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre> cc [flag...] file... -lcurses [library...] #include<curses.h> int addwchstr(chtype *wchstr); int addwchnstr(chtype *wchstr, int n); int waddwchstr(WINDOW *win, chtype *wchstr); int waddwchnstr(WINDOW *win, chtype *wchstr, int n); int mvaddwchstr(int y, int x, chtype *wchstr); int mvaddwchnstr(int y, int x, chtype *wchstr, int n); int mvwaddwchstr(WINDOW *win, int y, int x, chtype *wchstr); int mvwaddwchnstr(WINDOW *win, int y, int x, chtype *wchstr, int n); </pre> | | | | |
| DESCRIPTION | <p> All of these routines copy <code>wchstr</code>, which points to a string of <code>wchar_t</code> characters, directly into the window image structure starting at the current cursor position. The four routines with <code>n</code> as the last argument copy at most <code>n</code> elements, but no more than will fit on the line. If <code>n=-1</code> then the whole string is copied, to the maximum number that fit on the line. </p> <p> The position of the window cursor is not advanced. These routines work faster than <code>waddnwstr(3CURSES)</code> because they merely copy <code>wchstr</code> into the window image structure. On the other hand, care must be taken when using these functions because they don't perform any kind of checking (such as for the newline character), they do not advance the current cursor position, and they truncate the string, rather than wrapping it around to the new line. </p> | | | | |
| RETURN VALUE | <p> All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion, unless otherwise noted in the preceding routine descriptions. </p> | | | | |
| ATTRIBUTES | <p> See <code>attributes(5)</code> for descriptions of the following attributes: </p> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">MT-Level</td> <td style="text-align: center;">Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <p> <code>curses(3CURSES)</code>, <code>waddnwstr(3CURSES)</code>, <code>attributes(5)</code> </p> | | | | |
| NOTES | <p> The header file <code><curses.h></code> automatically includes the header files <code><stdio.h></code>, <code><unctrl.h></code> and <code><widec.h></code>. </p> | | | | |

`mvwaddwchstr(3CURSES)`

Note that all routines except `waddwchstr()` may be macros.

None of these routines can use the color attribute in `chtype`.

mvwadd_wchstr(3XCURSES)

| | | | | | | | |
|---------------------|--|---------------------|---|----------------|--|----------------|---|
| NAME | add_wchnstr, add_wchstr, mvadd_wchnstr, mvadd_wchstr, mvwadd_wchnstr, mvwadd_wchstr, wadd_wchnstr, wadd_wchstr – copy a string of complex characters (with renditions) to a window | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int add_wchnstr(const cchar_t *wchstr, int n); int add_wchstr(const cchar_t *wchstr); int mvadd_wchnstr(int y, int x, const cchar_t *wchstr, int n); int mvadd_wchstr(int y, int x, const cchar_t *wchstr); int mvwadd_wchnstr(WINDOW *win, int y, int x, const cchar_t *wchstr, int n); int mvwadd_wchstr(WINDOW *win, int y, int x, const cchar_t *wchstr); int wadd_wchstr(WINDOW *win, const cchar_t *wchstr); int wadd_wchnstr(WINDOW *win, const cchar_t *wchstr, int n);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>add_wchstr()</code> function copies the string of <code>cchar_t</code> characters to the <code>stdscr</code> window at the current cursor position. The <code>mvadd_wchstr()</code> and <code>mvwadd_wchstr()</code> functions copy the string to the starting position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <code>win</code>). The <code>wadd_wchstr()</code> is identical to <code>add_wchstr()</code>, but writes to the window specified by <code>win</code>.</p> <p>The <code>add_wchnstr()</code>, <code>wadd_wchnstr()</code>, <code>mvadd_wchnstr()</code>, and <code>mvwadd_wchnstr()</code> functions write <code>n</code> characters to the window, or as many as will fit on the line. If <code>n</code> is less than 0, the entire string is written, or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <code>x</code> and <code>y</code> parameters.</p> <p>These functions differ from the <code>addwstr(3XCURSES)</code> set of functions in two important respects. First, these functions do <i>not</i> advance the cursor after writing the string to the window. Second, the current window rendition (that is, the combination of attributes and color pair) is not combined with the character; only those attributes that are already part of the <code>cchar_t</code> character are used.</p> | | | | | | |
| PARAMETERS | <table><tr><td><code>wchstr</code></td><td>Is a pointer to the <code>cchar_t</code> string to be copied to the window.</td></tr><tr><td><code>n</code></td><td>Is the maximum number of characters to be copied from <code>wchstr</code>. If <code>n</code> is less than 0, the entire string is written or as much of it as fits on the line.</td></tr><tr><td><code>y</code></td><td>Is the <code>y</code> (row) coordinate of the starting position of <code>wchstr</code> in the window.</td></tr></table> | <code>wchstr</code> | Is a pointer to the <code>cchar_t</code> string to be copied to the window. | <code>n</code> | Is the maximum number of characters to be copied from <code>wchstr</code> . If <code>n</code> is less than 0, the entire string is written or as much of it as fits on the line. | <code>y</code> | Is the <code>y</code> (row) coordinate of the starting position of <code>wchstr</code> in the window. |
| <code>wchstr</code> | Is a pointer to the <code>cchar_t</code> string to be copied to the window. | | | | | | |
| <code>n</code> | Is the maximum number of characters to be copied from <code>wchstr</code> . If <code>n</code> is less than 0, the entire string is written or as much of it as fits on the line. | | | | | | |
| <code>y</code> | Is the <code>y</code> (row) coordinate of the starting position of <code>wchstr</code> in the window. | | | | | | |

`mvwadd_wchstr(3XCURSES)`

x Is the *x* (column) coordinate of the starting position of *wchstr* in the window.

win Is a pointer to the window to which the string is to be copied.

RETURN VALUES On success, these functions return `OK`. Otherwise, they return `ERR`.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addnwstr(3XCURSES)`, `add_wch(3XCURSES)`, `attr_off(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

mvwaddwstr(3CURSES)

| NAME | <code> curs_ addwstr, addwstr, addnwstr, waddwstr, waddnwstr, mvaddwstr, mvaddnwstr, mvwaddwstr, mvwaddnwstr </code> – add a string of <code>wchar_t</code> characters to a curses window and advance cursor | | | | |
|---------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lcurses [library...] #include<curses.h> int addwstr(wchar_t *wstr); int addnwstr(wchar_t *wstr, int n); int waddwstr(WINDOW *win, wchar_t *wstr); int waddnwstr(WINDOW *win, wchar_t *wstr, int n); int mvaddwstr(int y, int x, wchar_t *wstr); int mvaddnwstr(int y, int x, wchar_t *wstr, int n); int mvwaddwstr(WINDOW *win, int y, int x, wchar_t *wstr); int mvwaddnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);</pre> | | | | |
| DESCRIPTION | All of these routines write all the characters of the null-terminated <code>wchar_t</code> character string <code>wstr</code> on the given window. The effect is similar to calling <code>waddwch(3CURSES)</code> once for each <code>wchar_t</code> character in the string. The four routines with <code>n</code> as the last argument write at most <code>n</code> <code>wchar_t</code> characters. If <code>n</code> is negative, then the entire string will be added. | | | | |
| RETURN VALUE | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>waddwch(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header file <code><curses.h></code> automatically includes the header files <code><stdio.h></code> , <code><nctrl.h></code> and <code><widec.h></code> . Note that all of these routines except <code>waddwstr()</code> and <code>waddnwstr()</code> may be macros. | | | | |

| | |
|--------------------|---|
| NAME | addnwstr, addwstr, mvaddnwstr, mvaddwstr, mvwaddnwstr, mvwaddwstr, waddnwstr, waddwstr – add a wide-character string to a window |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addnwstr(const wchar_t *wstr, int n); int addwstr(const wchar_t *wstr); int mvaddnwstr(int y, int x, const wchar_t *wstr, int n); int mvaddwstr(int y, int x, const wchar_t *wstr); int mvwaddnwstr(WINDOW*win, int y, int x, const wchar_t *wstr, int n); int mvwaddwstr(WINDOW*win, int y, int x, const wchar_t *wstr); int waddnwstr(WINDOW*win, const wchar_t *wstr, int n); int waddwstr(WINDOW*win, const wchar_t *wstr);</pre> |
| DESCRIPTION | <p>The <code>addwstr()</code> function writes a null-terminated wide-character string to the <code>stdscr</code> window at the current cursor position. The <code>waddwstr()</code> function performs an identical action, but writes the string to the window specified by <code>win</code>. The <code>mvaddwstr()</code> and <code>mvwaddwstr()</code> functions write the string to the position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <code>win</code>).</p> <p>The <code>addnwstr()</code>, <code>waddnwstr()</code>, <code>mvaddnwstr()</code>, and <code>mvwaddnwstr()</code> functions write at most <code>n</code> characters to the window. If <code>n</code> is less than 0, the entire string is written. The former two functions place the characters at the current cursor position; the latter two commands use the position specified by the <code>x</code> and <code>y</code> parameters.</p> <p>All of these functions advance the cursor after writing the string.</p> <p>These functions are functionally equivalent to building a <code>cchar_t</code> from the <code>wchar_t</code> and the window rendition (or background character and rendition) and calling the <code>wadd_wch(3XCURSES)</code> function once for each <code>wchar_t</code> in the string. Refer to the <code>curses(3XCURSES)</code> man page for a complete description of special character handling and of the interaction between the window rendition (or background character and rendition) and the character written.</p> <p>Note that these functions differ from the <code>add_wchnstr(3XCURSES)</code> set of functions in that the latter copy the string as is (without combining each character with the foreground and background attributes of the window).</p> |
| PARAMETERS | <p><code>wstr</code> Is a pointer to the wide-character string that is to be written to the window.</p> |

mvwaddwstr(3XCURSES)

- n* Is the maximum number of characters to be copied from *wstr*. If *n* is less than 0, the entire string is written or as much of it as fits on the line.
- y* Is the y (row) coordinate of the starting position of *wstr* in the window.
- x* Is the x (column) coordinate of the starting position of *wstr* in the window.
- win* Is a pointer to the window in which the string is to be written.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `add_wch(3XCURSES)`, `add_wchnstr(3XCURSES)`, `curses(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| | | | | | | | | | | | | | | | |
|----------------------|---|----------|---|-------------|--|--------------|---|-------------|---|----------|---|----------|---|------------|---|
| NAME | chgat, mvchgat, mvwchgat, wchgat – change the rendition of characters in a window | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int chgat(int <i>n</i>, attr_t <i>attr</i>, short <i>color</i>, const void *<i>opts</i>); int mvchgat(int <i>y</i>, int <i>x</i>, int <i>n</i>, attr_t <i>attr</i>, short <i>color</i>, const void *<i>opts</i>); int mvwchgat(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>, int <i>n</i>, attr_t <i>attr</i>, short <i>color</i>, const void *<i>opts</i>); int wchgat(WINDOW *<i>win</i>, int <i>n</i>, attr_t <i>attr</i>, short <i>color</i>, const void *<i>opts</i>);</pre> | | | | | | | | | | | | | | |
| DESCRIPTION | <p>These functions change the renditions of the next <i>n</i> characters in the current or specified window (or of the remaining characters on the current or specified line, if <i>n</i> is -1), beginning at the current or specified cursor position. The attributes and colors are specified by <i>attr</i> and <i>color</i> as for <code>setcchar(3XCURSES)</code>.</p> <p>These function neither update the cursor nor perform wrapping.</p> <p>A value of <i>n</i> that is greater than the remaining characters on a line is not an error.</p> <p>The <i>opts</i> argument is reserved for definition in a future release. Currently, the application must provide a null pointer for <i>opts</i>.</p> | | | | | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>n</i></td> <td>Is the number of characters whose rendition is to be changed.</td> </tr> <tr> <td><i>attr</i></td> <td>Is the set of attributes to be assigned to the characters.</td> </tr> <tr> <td><i>color</i></td> <td>Is the new color pair to be assigned to the characters.</td> </tr> <tr> <td><i>opts</i></td> <td>Is reserved for future use. Currently, this must be a null pointer.</td> </tr> <tr> <td><i>y</i></td> <td>Is the y (row) coordinate of the starting position in the window.</td> </tr> <tr> <td><i>x</i></td> <td>Is the x (column) coordinate of the starting position in the window. changed in the window.</td> </tr> <tr> <td><i>win</i></td> <td>Is a pointer to the window in which the rendition of characters is to be changed.</td> </tr> </table> | <i>n</i> | Is the number of characters whose rendition is to be changed. | <i>attr</i> | Is the set of attributes to be assigned to the characters. | <i>color</i> | Is the new color pair to be assigned to the characters. | <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | <i>y</i> | Is the y (row) coordinate of the starting position in the window. | <i>x</i> | Is the x (column) coordinate of the starting position in the window. changed in the window. | <i>win</i> | Is a pointer to the window in which the rendition of characters is to be changed. |
| <i>n</i> | Is the number of characters whose rendition is to be changed. | | | | | | | | | | | | | | |
| <i>attr</i> | Is the set of attributes to be assigned to the characters. | | | | | | | | | | | | | | |
| <i>color</i> | Is the new color pair to be assigned to the characters. | | | | | | | | | | | | | | |
| <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | | | | | | | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of the starting position in the window. | | | | | | | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of the starting position in the window. changed in the window. | | | | | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the rendition of characters is to be changed. | | | | | | | | | | | | | | |
| RETURN VALUES | Upon successful completion, these functions returned OK. Otherwise, they return ERR. | | | | | | | | | | | | | | |
| ERRORS | No errors are defined. | | | | | | | | | | | | | | |

mvwchgat(3XCURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `bkgrnd(3XCURSES)`, `libcurses(3XCURSES)`, `setcchar(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_delch, delch, wdelch, mvdelch, mvwdelch – delete character under cursor in a curses window | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int delch(void); int wdelch(WINDOW *win); int mvdelch(int y, int x); int mvwdelch(WINDOW *win, int y, int x);</pre> | | | | |
| DESCRIPTION | With these routines the character under the cursor in the window is deleted; all characters to the right of the cursor on the same line are moved to the left one position and the last character on the line is filled with a blank. The cursor position does not change (after moving to <i>y, x</i> , if specified). This does not imply use of the hardware delete character feature. | | | | |
| RETURN VALUES | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), attributes(5) | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that <code>delch()</code>, <code>mvdelch()</code>, and <code>mvwdelch()</code> may be macros.</p> | | | | |

mvwdelch(3XCURSES)

NAME delch, mvdelch, mvwdelch, wdelch – remove a character

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

int delch(void);

int mvdelch(int y, int x);

int mvwdelch(WINDOW *win, int y, int x);

int wdelch(WINDOW *win);
```

DESCRIPTION The delch() and wdelch() functions delete the character at the current cursor position from stdscr and win, respectively. All remaining characters after cursor through to the end of the line are shifted one character towards the start of the line. The last character on the line becomes a space; characters on other lines are not affected.

The mvdelch() and mvwdelch() functions delete the character at the position specified by the x and y parameters; the former deletes the character from stdscr; the latter from win.

PARAMETERS

y Is the y (row) coordinate of the position of the character to be removed.

x Is the x (column) coordinate of the position of the character to be removed.

win Is a pointer to the window containing the character to be removed.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO bkgdset(3XCURSES), insch(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

| NAME | curs_getch, getch, wgetch, mvwgetch, mvwgetch, ungetch – get (or push back) characters from curses terminal keyboard | | | | |
|----------------------|---|------|----------|-----------|-----------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int getch(void); int wgetch(WINDOW *win); int mvwgetch(int y, int x); int mvwgetch(WINDOW *win, int y, int x); int ungetch(int ch);</pre> | | | | |
| DESCRIPTION | <p>With the <code>getch()</code>, <code>wgetch()</code>, <code>mvwgetch()</code>, and <code>mvwgetch()</code> routines a character is read from the terminal associated with the window. In no-delay mode, if no input is waiting, the value <code>ERR</code> is returned. In delay mode, the program waits until the system passes text through to the program. Depending on the setting of <code>cbreak()</code>, this is after one character (<code>cbreak</code> mode), or after the first newline (<code>nocbreak</code> mode). In half-delay mode, the program waits until a character is typed or the specified timeout has been reached. Unless <code>noecho()</code> has been set, the character will also be echoed into the designated window.</p> <p>If the window is not a pad, and it has been moved or modified since the last call to <code>wrefresh()</code>, <code>wrefresh()</code> will be called before another character is read.</p> <p>If <code>keypad()</code> is <code>TRUE</code>, and a function key is pressed, the token for that function key is returned instead of the raw characters. Possible function keys are defined in <code><curses.h></code> with integers beginning with <code>0401</code>, whose names begin with <code>KEY_</code>. If a character that could be the beginning of a function key (such as escape) is received, <code>curses</code> sets a timer. If the remainder of the sequence does not come in within the designated time, the character is passed through; otherwise, the function key value is returned. For this reason, many terminals experience a delay between the time a user presses the escape key and the escape is returned to the program. Since tokens returned by these routines are outside the ASCII range, they are not printable.</p> <p>The <code>ungetch()</code> routine places <code>ch</code> back onto the input queue to be returned by the next call to <code>wgetch()</code>.</p> | | | | |
| Function Keys | <p>The following function keys, defined in <code><curses.h></code>, might be returned by <code>getch()</code> if <code>keypad()</code> has been enabled. Note that not all of these may be supported on a particular terminal if the terminal does not transmit a unique code when the key is pressed or if the definition for the key is not present in the <code>terminfo</code> database.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Name</th> <th style="text-align: left; padding: 2px;">Key name</th> </tr> </thead> <tbody> <tr> <td style="padding: 2px;">KEY_BREAK</td> <td style="padding: 2px;">Break key</td> </tr> </tbody> </table> | Name | Key name | KEY_BREAK | Break key |
| Name | Key name | | | | |
| KEY_BREAK | Break key | | | | |

mvwgetch(3CURSES)

| <i>Name</i> | <i>Key name</i> |
|-------------------|---|
| KEY_DOWN | The four arrow keys . . . |
| KEY_UP | |
| KEY_LEFT | |
| KEY_RIGHT | |
| KEY_HOME | Home key (upward+left arrow) |
| KEY_BACKSPACE | Backspace |
| KEY_F0 | Function keys; space for 64 keys is reserved. |
| KEY_F(<i>n</i>) | For $0 \leq n \leq 63$ |
| KEY_DL | Delete line |
| KEY_IL | Insert line |
| KEY_DC | Delete character |
| KEY_IC | Insert char or enter insert mode |
| KEY_EIC | Exit insert char mode |
| KEY_CLEAR | Clear screen |
| KEY_EOS | Clear to end of screen |
| KEY_EOL | Clear to end of line |
| KEY_SF | Scroll 1 line forward |
| KEY_SR | Scroll 1 line backward (reverse) |
| KEY_NPAGE | Next page |
| KEY_PPAGE | Previous page |
| KEY_STAB | Set tab |
| KEY_CTAB | Clear tab |
| KEY_CATAB | Clear all tabs |
| KEY_ENTER | Enter or send |
| KEY_SRESET | Soft (partial) reset |
| KEY_RESET | Reset or hard reset |
| KEY_PRINT | Print or copy |
| KEY_LL | Home down or bottom (lower left). Keypad is arranged like this: (Row 1) A1 up A3 (Row 2) left B2 right (Row 3) C1 down C3 |

| <i>Name</i> | <i>Key name</i> |
|---------------|-----------------------|
| KEY_A1 | Upper left of keypad |
| KEY_A3 | Upper right of keypad |
| KEY_B2 | Center of keypad |
| KEY_C1 | Lower left of keypad |
| KEY_C3 | Lower right of keypad |
| KEY_BTAB | Back tab key |
| KEY_BEG | Beg(inning) key |
| KEY_CANCEL | Cancel key |
| KEY_CLOSE | Close key |
| KEY_COMMAND | Cmd (command) key |
| KEY_COPY | Copy key |
| KEY_CREATE | Create key |
| KEY_END | End key |
| KEY_EXIT | Exit key |
| KEY_FIND | Find key |
| KEY_HELP | Help key |
| KEY_MARK | Mark key |
| KEY_MESSAGE | Message key |
| KEY_MOVE | Move key |
| KEY_NEXT | Next object key |
| KEY_OPEN | Open key |
| KEY_OPTIONS | Options key |
| KEY_PREVIOUS | Previous object key |
| KEY_REDO | Redo key |
| KEY_REFERENCE | Reference key |
| KEY_REFRESH | Refresh key |
| KEY_REPLACE | Replace key |
| KEY_RESTART | Restart key |
| KEY_RESUME | Resume key |

mvwgetch(3CURSES)

| <i>Name</i> | <i>Key name</i> |
|---------------|-------------------------|
| KEY_SAVE | Save key |
| KEY_SBEG | Shifted beginning key |
| KEY_SCANCEL | Shifted cancel key |
| KEY_SCOMMAND | Shifted command key |
| KEY_SCOPY | Shifted copy key |
| KEY_SCREATE | Shifted create key |
| KEY_SDC | Shifted delete char key |
| KEY_SDL | Shifted delete line key |
| KEY_SELECT | Select key |
| KEY_SEND | Shifted end key |
| KEY_SEOL | Shifted clear line key |
| KEY_SEXIT | Shifted exit key |
| KEY_SFIND | Shifted find key |
| KEY_SHELP | Shifted help key |
| KEY_SHOME | Shifted home key |
| KEY_SIC | Shifted input key |
| KEY_SLEFT | Shifted left arrow key |
| KEY_SMESSAGE | Shifted message key |
| KEY_SMOVE | Shifted move key |
| KEY_SNEXT | Shifted next key |
| KEY_SOPTIONS | Shifted options key |
| KEY_SPREVIOUS | Shifted prev key |
| KEY_SPRINT | Shifted print key |
| KEY_SREDO | Shifted redo key |
| KEY_SREPLACE | Shifted replace key |
| KEY_SRIGHT | Shifted right arrow |
| KEY_SRSUME | Shifted resume key |
| KEY_SSAVE | Shifted save key |
| KEY_SSUSPEND | Shifted suspend key |

| <i>Name</i> | <i>Key name</i> |
|-------------|------------------|
| KEY_SUNDO | Shifted undo key |
| KEY_SUSPEND | Suspend key |
| KEY_UNDO | Undo key |

RETURN VALUES All routines return the integer `ERR` upon failure. The `ungetch()` routine returns an integer value other than `ERR` upon successful completion. The other routines return the next input character or function key code upon successful completion.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_inopts(3CURSES)`, `curs_move(3CURSES)`, `curs_refresh(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Use of the escape key for a single character function is discouraged.

When using `getch()`, `wgetch()`, `mvgetch()`, or `mvwgetch()`, `nocbreak` mode (`nocbreak()`) and `echo` mode (`echo()`) should not be used at the same time. Depending on the state of the tty driver when each character is typed, the program may produce undesirable results.

Note that `getch()`, `mvgetch()`, and `mvwgetch()` may be macros.

mvwgetch(3XCURSES)

| | | | | | |
|--|---|-------------|---|---------------|---|
| NAME | getch, wgetch, mvgetch, mvwgetch – get a single-byte character from the terminal | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int getch(void); int wgetch(WINDOW *win); int mvgetch(int y, int x); int mvwgetch(WINDOW *win, int y, int x);</pre> | | | | |
| PARAMETERS | <p><i>win</i> Is a pointer to the window associated with the terminal from which the character is to be read.</p> <p><i>y</i> Is the y (row) coordinate for the position of the character to be read.</p> <p><i>x</i> Is the x (column) coordinate for the position of the character to be read.</p> | | | | |
| DESCRIPTION | <p>These functions read a single-byte character from the terminal associated with the current or specified window. The results are unspecified if the input is not a single-byte character. If keypad(3XCURSES) is enabled, these functions respond to the pressing of a function key by returning the corresponding KEY_ value defined in <curses.h></p> <p>Processing of terminal input is subject to the general rules described on the keypad(3XCURSES) manual page.</p> <p>If echoing is enabled, then the character is echoed as though it were provided as an input argument to addch(3XCURSES), except for the following characters:</p> <table><tr><td><backspace></td><td>The input is interpreted as follows: unless the cursor already was in column 0, <backspace> moves the cursor one column toward the start of the current line and any characters after the <backspace> are added or inserted starting there. The character at the resulting cursor position is then deleted as though delch(3XCURSES) were called, except that if the cursor was originally in the first column of the line, the user is alerted as though beep(3XCURSES) were called.</td></tr><tr><td>Function keys</td><td>The user is alerted as though beep() were called. Information concerning the function keys is not returned to the caller.</td></tr></table> <p>If the current or specified window is not a pad, and it has been moved modified since the last refresh operation, then it will be refreshed before another character is read.</p> | <backspace> | The input is interpreted as follows: unless the cursor already was in column 0, <backspace> moves the cursor one column toward the start of the current line and any characters after the <backspace> are added or inserted starting there. The character at the resulting cursor position is then deleted as though delch(3XCURSES) were called, except that if the cursor was originally in the first column of the line, the user is alerted as though beep(3XCURSES) were called. | Function keys | The user is alerted as though beep() were called. Information concerning the function keys is not returned to the caller. |
| <backspace> | The input is interpreted as follows: unless the cursor already was in column 0, <backspace> moves the cursor one column toward the start of the current line and any characters after the <backspace> are added or inserted starting there. The character at the resulting cursor position is then deleted as though delch(3XCURSES) were called, except that if the cursor was originally in the first column of the line, the user is alerted as though beep(3XCURSES) were called. | | | | |
| Function keys | The user is alerted as though beep() were called. Information concerning the function keys is not returned to the caller. | | | | |
| Constant Values for Function Keys | The following is a list of tokens for function keys that are returned by the getch() set of functions if keypad handling is enabled (some terminals may not support all tokens). | | | | |

| Constant | Description |
|-------------------|---|
| KEY_BREAK | Break key |
| KEY_DOWN | The down arrow key |
| KEY_UP | The up arrow key |
| KEY_LEFT | The left arrow key |
| KEY_RIGHT | The right arrow key |
| KEY_HOME | Home key |
| KEY_BACKSPACE | Backspace |
| KEY_F0 | Function keys. Space for 64 keys is reserved. |
| KEY_F(<i>n</i>) | For $0 \leq n \leq 63$ |
| KEY_DL | Delete line |
| KEY_IL | Insert line |
| KEY_DC | Delete character |
| KEY_IC | Insert char or enter insert mode |
| KEY_EIC | Exit insert char mode |
| KEY_CLEAR | Clear screen |
| KEY_EOS | Clear to end of screen |
| KEY_EOL | Clear to end of line |
| KEY_SF | Scroll 1 line forward |
| KEY_SR | Scroll 1 line backwards |
| KEY_NPAGE | Next page |
| KEY_PPAGE | Previous page |
| KEY_STAB | Set tab |
| KEY_CTAB | Clear tab |
| KEY_CATAB | Clear all tabs |
| KEY_ENTER | Enter or send |
| KEY_SRESET | Soft (partial) reset |
| KEY_RESET | Reset or hard reset |
| KEY_PRINT | Print or copy |
| KEY_LL | Home down or bottom (lower left) |

mvwgetch(3XCURSES)

| Constant | Description |
|---------------|-----------------------|
| KEY_A1 | Upper left of keypad |
| KEY_A3 | Upper right of keypad |
| KEY_B2 | Center of keypad |
| KEY_C1 | Lower left of keypad |
| KEY_C3 | Lower right of keypad |
| KEY_BTAB | Back tab |
| KEY_BEG | Beginning key |
| KEY_CANCEL | Cancel key |
| KEY_CLOSE | Close key |
| KEY_COMMAND | Cmd (command) key |
| KEY_COPY | Copy key |
| KEY_CREATE | Create key |
| KEY_END | End key |
| KEY_EXIT | Exit key |
| KEY_FIND | Find key |
| KEY_HELP | Help key |
| KEY_MARK | Mark key |
| KEY_MESSAGE | Message key |
| KEY_MOVE | Move key |
| KEY_NEXT | Next object key |
| KEY_OPEN | Open key |
| KEY_OPTIONS | Options key |
| KEY_PREVIOUS | Previous object key |
| KEY_REDO | Redo key |
| KEY_REFERENCE | Reference key |
| KEY_REFRESH | Refresh key |
| KEY_REPLACE | Replace key |
| KEY_RESTART | Restart key |
| KEY_RESUME | Resume key |

| Constant | Description |
|---------------|-------------------------|
| KEY_SAVE | Save key |
| KEY_SBEG | Shifted beginning key |
| KEY_SCANCEL | Shifted cancel key |
| KEY_SCOMMAND | Shifted command key |
| KEY_SCOPY | Shifted copy key |
| KEY_SCREATE | Shifted create key |
| KEY_SDC | Shifted delete char key |
| KEY_SDL | Shifted delete line key |
| KEY_SELECT | Select key |
| KEY_SEND | Shifted end key |
| KEY_SEOL | Shifted clear line key |
| KEY_SEXIT | Shifted exit key |
| KEY_SFIND | Shifted find key |
| KEY_SHELP | Shifted help key |
| KEY_SHOME | Shifted home key |
| KEY_SIC | Shifted input key |
| KEY_SLEFT | Shifted left arrow key |
| KEY_SMESSAGES | Shifted messages key |
| KEY_SMOVE | Shifted move key |
| KEY_SNEXT | Shifted next key |
| KEY_SOPTIONS | Shifted options key |
| KEY_SPREVIOUS | Shifted previous key |
| KEY_SPRINT | Shifted print key |
| KEY_SREDO | Shifted redo key |
| KEY_SREPLACE | Shifted replace key |
| KEY_SRIGHT | Shifted right arrow key |
| KEY_SRSUME | Shifted resume key |
| KEY_SSAVE | Shifted save key |
| KEY_SSUSPEND | Shifted suspend key |

mvwgetch(3XCURSES)

| Constant | Description |
|-------------|------------------|
| KEY_SUNDO | Shifted undo key |
| KEY_SUSPEND | Suspend key |
| KEY_UNDO | Undo key |

RETURN VALUES Upon successful completion, these functions return the single-byte character, `KEY_` value, or `ERR`. When in the nodelay mode and no data is available, `ERR` is returned.

ERRORS No errors are defined.

USAGE Applications should not define the escape key by itself as a single-character function. When using these functions, `nocbreak` mode (`cbreak(3XCURSES)`) and `echo` mode (`echo(3XCURSES)`) should not be used at the same time. Depending on the state of the terminal when each character is typed, the application may produce undesirable results.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `cbreak(3XCURSES)`, `echo(3XCURSES)`, `halfdelay(3XCURSES)`, `keypad(3XCURSES)`, `libcurses(3XCURSES)`, `nodelay(3XCURSES)`, `notimeout(3XCURSES)`, `raw(3XCURSES)`, `timeout(3XCURSES)`, `attributes(5)`, `standards(5)`

| | | | | | | | | | | | |
|--------------------|--|------------|--|----------|---|----------|--|----------|---|------------|---|
| NAME | getnstr, getstr, mvgetnstr, mvgetstr, mvwgetnstr, mvwgetstr, wgetnstr, wgetstr – get a multibyte character string from terminal | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int getnstr(char *str, int n); int getstr(char *str); int mvgetnstr(int y, int x, char *str, int n); int mvgetstr(int y, int x, char *str); int mvwgetnstr(WINDOW *win, int y, int x, char *str, int n); int mvwgetstr(WINDOW *win, int y, int x, char *str); int wgetnstr(WINDOW *win, char *str, int n); int wgetstr(WINDOW *win, char *str);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The <code>getstr()</code> and <code>wgetstr()</code> functions get a character string from the terminal associated with the window <code>stdscr</code> or window <code>win</code>, respectively. The <code>mvgetstr()</code> and <code>mvwgetstr()</code> functions move the cursor to the position specified in <code>stdscr</code> or <code>win</code>, respectively, then get a character string.</p> <p>These functions call <code>wgetch(3XCURSES)</code> and place each received character in <code>str</code> until a newline is received, which is also placed in <code>str</code>. The erase and kill characters set by the user are processed.</p> <p>The <code>getnstr()</code>, <code>mvgetnstr()</code>, <code>mvwgetnstr()</code> and <code>wgetnstr()</code> functions read at most <code>n</code> characters. These functions are used to prevent overflowing the input buffer.</p> <p>The <code>getnstr()</code>, <code>wgetnstr()</code>, <code>mvgetnstr()</code>, and <code>mvwgetnstr()</code> functions only return complete multibyte characters. If the area pointed to by <code>str</code> is not large enough to hold at least one character, these functions fail.</p> | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>str</i></td> <td>Is a pointer to the area where the character string is to be placed.</td> </tr> <tr> <td><i>n</i></td> <td>Is the maximum number of characters to read from input.</td> </tr> <tr> <td><i>y</i></td> <td>Is the y (row) coordinate of starting position of character string to be read.</td> </tr> <tr> <td><i>x</i></td> <td>Is the x (column) coordinate of starting position of character string to be read.</td> </tr> <tr> <td><i>win</i></td> <td>Points to the window associated with the terminal from which the character is to be read.</td> </tr> </table> | <i>str</i> | Is a pointer to the area where the character string is to be placed. | <i>n</i> | Is the maximum number of characters to read from input. | <i>y</i> | Is the y (row) coordinate of starting position of character string to be read. | <i>x</i> | Is the x (column) coordinate of starting position of character string to be read. | <i>win</i> | Points to the window associated with the terminal from which the character is to be read. |
| <i>str</i> | Is a pointer to the area where the character string is to be placed. | | | | | | | | | | |
| <i>n</i> | Is the maximum number of characters to read from input. | | | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of starting position of character string to be read. | | | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of starting position of character string to be read. | | | | | | | | | | |
| <i>win</i> | Points to the window associated with the terminal from which the character is to be read. | | | | | | | | | | |

mvwgetnstr(3XCURSES)

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `getch(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

NAME | curs_getwstr, getwstr, getnwstr, wgetwstr, wgetnwstr, mvgetwstr, mvgetnwstr, mvwgetwstr, mvwgetnwstr – get wchar_t character strings from curses terminal keyboard

SYNOPSIS |

```
cc [ flag ... ] file ... -lcurses [ library .. ]
#include <curses.h>

int getwstr(wchar_t *wstr);
int getnwstr(wchar_t *wstr, int n);
int wgetwstr(WINDOW *win, wchar_t *wstr);
int wgetnwstr(WINDOW *win, wchar_t *wstr, int n);
int mvgetwstr(int y, int x, wchar_t *wstr);
int mvgetnwstr(int y, int x, wchar_t *wstr, int n);
int mvwgetwstr(WINDOW *win, int y, int x, wchar_t *wstr);
int mvwgetnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);
```

DESCRIPTION | The effect of getwstr() is as though a series of calls to getch(3CURSES) were made, until a newline and carriage return is received. The resulting value is placed in the area pointed to by the wchar_t pointer wstr. getnwstr() reads at most n wchar_t characters, thus preventing a possible overflow of the input buffer. The user's erase and kill characters are interpreted, as well as any special keys (such as function keys, HOME key, CLEAR key, etc.).

RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES | See attributes(5) for a description of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | curses(3CURSES), getch(3CURSES), attributes(5)

NOTES | The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h>, and <wctype.h>.

Note that all routines except wgetnwstr() may be macros.

mvwgetn_wstr(3XCURSES)

| | | | | | | | | | | | |
|----------------------|---|-------------|--|----------|---|----------|--|----------|---|------------|---|
| NAME | <code>getn_wstr</code> , <code>get_wstr</code> , <code>mvgetn_wstr</code> , <code>mvget_wstr</code> , <code>mvwgetn_wstr</code> , <code>mvwget_wstr</code> , <code>wgetn_wstr</code> , <code>wget_wstr</code> – get a wide character string from terminal | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int getn_wstr(wint_t *wstr, int n); int get_wstr(wint_t *wstr); int mvgetn_wstr(int y, int x, wint_t *wstr, int n); int mvget_wstr(int y, int x, wint_t *wstr); int mvwgetn_wstr(WINDOW *win, int y, int x, wint_t *wstr, int n); int mvwget_wstr(WINDOW *win, int y, int x, wint_t *wstr); int wgetn_wstr(WINDOW *win, wint_t *wstr, int n); int wget_wstr(WINDOW *win, wint_t *wstr);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The <code>get_wstr()</code> and <code>wget_wstr()</code> functions get a wide character string from the terminal associated with the window <code>stdscr</code> or window <code>win</code>, respectively. The <code>mvget_wstr()</code> and <code>mvwget_wstr()</code> functions move the cursor to the position specified in <code>stdscr</code> or <code>win</code>, respectively, then get a wide character string.</p> <p>These functions call <code>wget_wch(3XCURSES)</code> and place each received character in <code>wstr</code> until a newline character, end-of-line character, or end-of-file character is received, which is also placed in <code>wstr</code>. The erase and kill characters set by the user are processed.</p> <p>The <code>getn_wstr()</code>, <code>mvgetn_wstr()</code>, <code>mvwgetn_wstr()</code> and <code>wgetn_wstr()</code> functions read at most <code>n</code> characters. These functions are used to prevent overflowing the input buffer.</p> | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>wstr</i></td><td>Is a pointer to the area where the character string is to be placed.</td></tr><tr><td><i>n</i></td><td>Is the maximum number of characters to read from input.</td></tr><tr><td><i>y</i></td><td>Is the y (row) coordinate of starting position of character string to be read.</td></tr><tr><td><i>x</i></td><td>Is the x (column) coordinate of starting position of character string to be read.</td></tr><tr><td><i>win</i></td><td>points to the window associated with the terminal from which the character is to be read.</td></tr></table> | <i>wstr</i> | Is a pointer to the area where the character string is to be placed. | <i>n</i> | Is the maximum number of characters to read from input. | <i>y</i> | Is the y (row) coordinate of starting position of character string to be read. | <i>x</i> | Is the x (column) coordinate of starting position of character string to be read. | <i>win</i> | points to the window associated with the terminal from which the character is to be read. |
| <i>wstr</i> | Is a pointer to the area where the character string is to be placed. | | | | | | | | | | |
| <i>n</i> | Is the maximum number of characters to read from input. | | | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of starting position of character string to be read. | | | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of starting position of character string to be read. | | | | | | | | | | |
| <i>win</i> | points to the window associated with the terminal from which the character is to be read. | | | | | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | | | | | |
| ERRORS | None. | | | | | | | | | | |

mvwgetn_wstr(3XCURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `get_wch(3XCURSES)`, `getnstr(3XCURSES)`, `libcurses(3XCURSES)`,
`attributes(5)`, `standards(5)`

mvwgetstr(3CURSES)

| NAME | <code> curs_getstr, getstr, wgetstr, mvgetstr, mvwgetstr, wgetnstr </code> – get character strings from curses terminal keyboard | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int getstr(char *<i>str</i>); int wgetstr(WINDOW *<i>win</i>, char *<i>str</i>); int mvgetstr(int <i>y</i>, int <i>x</i>, char *<i>str</i>); int mvwgetstr(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>, char *<i>str</i>); int wgetnstr(WINDOW *<i>win</i>, char *<i>str</i>, int <i>n</i>);</pre> | | | | |
| DESCRIPTION | The effect of <code>getstr()</code> is as though a series of calls to <code>getch()</code> were made, until a newline or carriage return is received. The resulting value is placed in the area pointed to by the character pointer <code>str</code> . <code>wgetnstr()</code> reads at most <code>n</code> characters, thus preventing a possible overflow of the input buffer. The user's erase and kill characters are interpreted, as well as any special keys (such as function keys, HOME key, and CLEAR key.) | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code> curs_getch(3CURSES), curses(3CURSES), attributes(5) </code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . Note that <code>getstr()</code> , <code>mvgetstr()</code> , and <code>mvwgetstr()</code> may be macros. | | | | |

| | | | | | | | | | | | |
|--------------------|--|------------|--|----------|---|----------|--|----------|---|------------|---|
| NAME | getnstr, getstr, mvgetnstr, mvgetstr, mvwgetnstr, mvwgetstr, wgetnstr, wgetstr – get a multibyte character string from terminal | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int getnstr(char *str, int n); int getstr(char *str); int mvgetnstr(int y, int x, char *str, int n); int mvgetstr(int y, int x, char *str); int mvwgetnstr(WINDOW *win, int y, int x, char *str, int n); int mvwgetstr(WINDOW *win, int y, int x, char *str); int wgetnstr(WINDOW *win, char *str, int n); int wgetstr(WINDOW *win, char *str);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The <code>getstr()</code> and <code>wgetstr()</code> functions get a character string from the terminal associated with the window <code>stdscr</code> or window <code>win</code>, respectively. The <code>mvgetstr()</code> and <code>mvwgetstr()</code> functions move the cursor to the position specified in <code>stdscr</code> or <code>win</code>, respectively, then get a character string.</p> <p>These functions call <code>wgetch(3XCURSES)</code> and place each received character in <code>str</code> until a newline is received, which is also placed in <code>str</code>. The erase and kill characters set by the user are processed.</p> <p>The <code>getnstr()</code>, <code>mvgetnstr()</code>, <code>mvwgetnstr()</code> and <code>wgetnstr()</code> functions read at most <code>n</code> characters. These functions are used to prevent overflowing the input buffer.</p> <p>The <code>getnstr()</code>, <code>wgetnstr()</code>, <code>mvgetnstr()</code>, and <code>mvwgetnstr()</code> functions only return complete multibyte characters. If the area pointed to by <code>str</code> is not large enough to hold at least one character, these functions fail.</p> | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>str</i></td> <td>Is a pointer to the area where the character string is to be placed.</td> </tr> <tr> <td><i>n</i></td> <td>Is the maximum number of characters to read from input.</td> </tr> <tr> <td><i>y</i></td> <td>Is the y (row) coordinate of starting position of character string to be read.</td> </tr> <tr> <td><i>x</i></td> <td>Is the x (column) coordinate of starting position of character string to be read.</td> </tr> <tr> <td><i>win</i></td> <td>Points to the window associated with the terminal from which the character is to be read.</td> </tr> </table> | <i>str</i> | Is a pointer to the area where the character string is to be placed. | <i>n</i> | Is the maximum number of characters to read from input. | <i>y</i> | Is the y (row) coordinate of starting position of character string to be read. | <i>x</i> | Is the x (column) coordinate of starting position of character string to be read. | <i>win</i> | Points to the window associated with the terminal from which the character is to be read. |
| <i>str</i> | Is a pointer to the area where the character string is to be placed. | | | | | | | | | | |
| <i>n</i> | Is the maximum number of characters to read from input. | | | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of starting position of character string to be read. | | | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of starting position of character string to be read. | | | | | | | | | | |
| <i>win</i> | Points to the window associated with the terminal from which the character is to be read. | | | | | | | | | | |

mvwgetstr(3XCURSES)

RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR.

ERRORS | None.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | `getch(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

NAME | curs_getwch, getwch, wgetwch, mvgetwch, mvwgetwch, ungetwch – get (or push back) wchar_t characters from curses terminal keyboard

SYNOPSIS | `cc [flag ...] file ... -lcurses [library ..]`
`#include <curses.h>`
`int getwch(void);`
`int wgetwch(WINDOW *win);`
`int mvgetwch(int y, int x);`
`int mvwgetwch(WINDOW *win, int y, int x);`
`int ungetwch(int wch);`

DESCRIPTION | The `getwch()`, `wgetwch()`, `mvgetwch()`, and `mvwgetwch()` routines read an EUC character from the terminal associated with the window, transform it into a `wchar_t` character, and return a `wchar_t` character. In no-delay mode, if no input is waiting, the value `ERR` is returned. In delay mode, the program waits until the system passes text through to the program. Depending on the setting of `cbreak`, this is after one character (`cbreak` mode), or after the first newline (`nocbreak` mode). In half-delay mode, the program waits until a character is typed or the specified timeout has been reached. Unless `noecho` has been set, the character will also be echoed into the designated window.

If the window is not a pad, and it has been moved or modified since the last call to `wrefresh(3CURSES)`, `wrefresh` will be called before another character is read.

If `keypad` is `TRUE`, and a function key is pressed, the token for that function key is returned instead of the raw characters. Possible function keys are defined in `<curses.h>` with integers beginning with `0401`, whose names begin with `KEY_`. If a character that could be the beginning of a function key (such as escape) is received, `curses(3CURSES)` sets a timer. If the remainder of the sequence does not come in within the designated time, the character is passed through; otherwise, the function key value is returned. For this reason, many terminals experience a delay between the time a user presses the escape key and the escape is returned to the program.

The `ungetwch()` routine places `wch` back onto the input queue to be returned by the next call to `wgetwch()`.

Function Keys | The following function keys, defined in `<curses.h>`, might be returned by `getwch()` if `keypad` has been enabled. Note that not all of these may be supported on a particular terminal if the terminal does not transmit a unique code when the key is pressed or if the definition for the key is not present in the `terminfo(4)` database.

| <i>Name</i> | <i>Key name</i> |
|-------------|-----------------|
| KEY_BREAK | Break key |

mvwgetwch(3CURSES)

| <i>Name</i> | <i>Key name</i> |
|-------------------|---|
| KEY_DOWN | The four arrow keys . . . |
| KEY_UP | |
| KEY_LEFT | |
| KEY_RIGHT | |
| KEY_HOME | Home key (upward+left arrow) |
| KEY_BACKSPACE | Backspace |
| KEY_F0 | Function keys; space for 64 keys is reserved. |
| KEY_F(<i>n</i>) | For $0 \leq n \leq 63$ |
| KEY_DL | Delete line |
| KEY_IL | Insert line |
| KEY_DC | Delete character |
| KEY_IC | Insert char or enter insert mode |
| KEY_EIC | Exit insert char mode |
| KEY_CLEAR | Clear screen |
| KEY_EOS | Clear to end of screen |
| KEY_EOL | Clear to end of line |
| KEY_SF | Scroll 1 line forward |
| KEY_SR | Scroll 1 line backward (reverse) |
| KEY_NPAGE | Next page |
| KEY_PPAGE | Previous page |
| KEY_STAB | Set tab |
| KEY_CTAB | Clear tab |
| KEY_CATAB | Clear all tabs |
| KEY_ENTER | Enter or send |
| KEY_SRESET | Soft (partial) reset |
| KEY_RESET | Reset or hard reset |
| KEY_PRINT | Print or copy |
| KEY_LL | Home down or bottom (lower left). Keypad is arranged like this: A1 up A3 left B2 right C1 down C3 |

| <i>Name</i> | <i>Key name</i> |
|---------------|-----------------------|
| KEY_A1 | Upper left of keypad |
| KEY_A3 | Upper right of keypad |
| KEY_B2 | Center of keypad |
| KEY_C1 | Lower left of keypad |
| KEY_C3 | Lower right of keypad |
| KEY_BTAB | Back tab key |
| KEY_BEG | Beg(inning) key |
| KEY_CANCEL | Cancel key |
| KEY_CLOSE | Close key |
| KEY_COMMAND | Cmd (command) key |
| KEY_COPY | Copy key |
| KEY_CREATE | Create key |
| KEY_END | End key |
| KEY_EXIT | Exit key |
| KEY_FIND | Find key |
| KEY_HELP | Help key |
| KEY_MARK | Mark key |
| KEY_MESSAGE | Message key |
| KEY_MOVE | Move key |
| KEY_NEXT | Next object key |
| KEY_OPEN | Open key |
| KEY_OPTIONS | Options key |
| KEY_PREVIOUS | Previous object key |
| KEY_REDO | Redo key |
| KEY_REFERENCE | Reference key |
| KEY_REFRESH | Refresh key |
| KEY_REPLACE | Replace key |
| KEY_RESTART | Restart key |
| KEY_RESUME | Resume key |

mvwgetwch(3CURSES)

| <i>Name</i> | <i>Key name</i> |
|---------------|-------------------------|
| KEY_SAVE | Save key |
| KEY_SBEG | Shifted beginning key |
| KEY_SCANCEL | Shifted cancel key |
| KEY_SCOMMAND | Shifted command key |
| KEY_SCOPY | Shifted copy key |
| KEY_SCREATE | Shifted create key |
| KEY_SDC | Shifted delete char key |
| KEY_SDL | Shifted delete line key |
| KEY_SELECT | Select key |
| KEY_SEND | Shifted end key |
| KEY_SEOL | Shifted clear line key |
| KEY_SEXIT | Shifted exit key |
| KEY_SFIND | Shifted find key |
| KEY_SHELP | Shifted help key |
| KEY_SHOME | Shifted home key |
| KEY_SIC | Shifted input key |
| KEY_SLEFT | Shifted left arrow key |
| KEY_SMESSAGE | Shifted message key |
| KEY_SMOVE | Shifted move key |
| KEY_SNEXT | Shifted next key |
| KEY_SOPTIONS | Shifted options key |
| KEY_SPREVIOUS | Shifted prev key |
| KEY_SPRINT | Shifted print key |
| KEY_SREDO | Shifted redo key |
| KEY_SREPLACE | Shifted replace key |
| KEY_SRIGHT | Shifted right arrow |
| KEY_SRSUME | Shifted resume key |
| KEY_SSAVE | Shifted save key |
| KEY_SSUSPEND | Shifted suspend key |

| <i>Name</i> | <i>Key name</i> |
|-------------|------------------|
| KEY_SUNDO | Shifted undo key |
| KEY_SUSPEND | Suspend key |
| KEY_UNDO | Undo key |

RETURN VALUE All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3CURSES)`, `curl_inopts(3CURSES)`, `curl_move(3CURSES)`, `wrefresh(3CURSES)`, `terminfo(4)`, `attributes(5)`

NOTES The header file `<curses.h>` automatically includes the header files `<stdio.h>`, `<unctrl.h>` and `<widec.h>`.

Use of the escape key by a programmer for a single character function is discouraged.

When using `getwch()`, `wgetwch()`, `mvgetwch()`, or `mvwgetwch()`, `nocbreak` mode and `echo` mode should not be used at the same time. Depending on the state of the tty driver when each character is typed, the program may produce undesirable results.

Note that `getwch()`, `mvgetwch()`, and `mvwgetwch()` may be macros.

mvwget_wch(3XCURSES)

| | |
|--------------------|---|
| NAME | get_wch, wget_wch, mvget_wch, mvwget_wch – get a wide character from terminal |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int get_wch(wint_t *ch); int wget_wch(WINDOW *win, wint_t *ch); int mvget_wch(int y, int x, wint_t *ch); int mvwget_wch(WINDOW *win, int y, int x, wint_t *ch);</pre> |
| DESCRIPTION | <p>The <code>get_wch()</code> and <code>wget_wch()</code> functions get a wide character from the terminal associated with the window <code>stdscr</code> or window <code>win</code>, respectively. The <code>mvget_wch()</code> and <code>mvwget_wch()</code> functions move the cursor to the position specified in <code>stdscr</code> or <code>win</code>, respectively, then get a character.</p> <p>If the window is not a pad and has been changed since the last call to <code>refresh(3XCURSES)</code>, <code>get_wch()</code> calls <code>refresh()</code> to update the window before the next character is read.</p> <p>The setting of certain functions affects the behavior of the <code>get_wch()</code> set of functions. For example, if <code>cbreak(3XCURSES)</code> is set, characters typed by the user are immediately processed. If <code>halfdelay(3XCURSES)</code> is set, <code>get_wch()</code> waits until a character is typed or returns <code>ERR</code> if no character is typed within the specified timeout period. This timeout can also be specified for individual windows with the <code>delay</code> parameter of <code>timeout(3XCURSES)</code>. A negative value waits for input; a value of 0 returns <code>ERR</code> if no input is ready; a positive value blocks until input arrives or the time specified expires (in which case <code>ERR</code> is returned). If <code>nodelay(3XCURSES)</code> is set, <code>ERR</code> is returned if no input is waiting; if not set, <code>get_wch()</code> waits until input arrives. Each character will be echoed to the window unless <code>noecho(3XCURSES)</code> has been set.</p> <p>If keypad handling is enabled (<code>keypad(3XCURSES)</code> is <code>TRUE</code>), the token for the function key (a <code>KEY_</code> value) is stored in the object pointed to by <code>ch</code> and <code>KEY_CODE_YES</code> is returned. If a character is received that could be the beginning of a function key (for example, <code>ESC</code>), an inter-byte timer is set. If the remainder of the sequence is not received before the time expires, the character is passed through; otherwise, the value of the function key is returned. If <code>notimeout()</code> is set, the inter-byte timer is not used.</p> <p>The <code>ESC</code> key is typically a prefix key used with function keys and should not be used as a single character.</p> <p>See the <code>getch(3XCURSES)</code> manual page for a list of tokens for function keys that are returned by the <code>get_wch()</code> set of functions if keypad handling is enabled (Some terminals may not support all tokens).</p> |

mvwget_wch(3XCURSES)

PARAMETERS

ch Is a pointer to a wide integer where the returned wide character or `KEY_` value can be stored.

win Is a pointer to the window associated with the terminal from which the character is to be read.

y Is the y (row) coordinate for the position of the character to be read.

x Is the x (column) coordinate for the position of the character to be read.

RETURN VALUES When these functions successfully report the pressing of a function key, they return `KEY_CODE_YES`. When they successfully report a wide character, they return `OK`. Otherwise, they return `ERR`.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `cbreak(3XCURSES)`, `echo(3XCURSES)`, `halfdelay(3XCURSES)`, `keypad(3XCURSES)`, `libcurses(3XCURSES)`, `nodelay(3XCURSES)`, `notimeout(3XCURSES)`, `raw(3XCURSES)`, `timeout(3XCURSES)`, `attributes(5)`, `standards(5)`

mvwgetwstr(3CURSES)

| NAME | <code>curs_getwstr</code> , <code>getwstr</code> , <code>getnwstr</code> , <code>wgetwstr</code> , <code>wgetnwstr</code> , <code>mvgetwstr</code> , <code>mvgetnwstr</code> , <code>mvwgetwstr</code> , <code>mvwgetnwstr</code> – get <code>wchar_t</code> character strings from curses terminal keyboard | | | | |
|---------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int getwstr(wchar_t *wstr); int getnwstr(wchar_t *wstr, int n); int wgetwstr(WINDOW *win, wchar_t *wstr); int wgetnwstr(WINDOW *win, wchar_t *wstr, int n); int mvgetwstr(int y, int x, wchar_t *wstr); int mvgetnwstr(int y, int x, wchar_t *wstr, int n); int mvwgetwstr(WINDOW *win, int y, int x, wchar_t *wstr); int mvwgetnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);</pre> | | | | |
| DESCRIPTION | The effect of <code>getwstr()</code> is as though a series of calls to <code>getwch(3CURSES)</code> were made, until a newline and carriage return is received. The resulting value is placed in the area pointed to by the <code>wchar_t</code> pointer <code>wstr</code> . <code>getnwstr()</code> reads at most <code>n</code> <code>wchar_t</code> characters, thus preventing a possible overflow of the input buffer. The user's erase and kill characters are interpreted, as well as any special keys (such as function keys, HOME key, CLEAR key, etc.). | | | | |
| RETURN VALUE | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for a description of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>getwch(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header file <code><curses.h></code> automatically includes the header files <code><stdio.h></code> , <code><unctrl.h></code> , and <code><widec.h></code> . Note that all routines except <code>wgetnwstr()</code> may be macros. | | | | |

| | |
|----------------------|---|
| NAME | getn_wstr, get_wstr, mvgetn_wstr, mvget_wstr, mvwgetn_wstr, mvwget_wstr, wgetn_wstr, wget_wstr – get a wide character string from terminal |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int getn_wstr(wint_t *wstr, int n); int get_wstr(wint_t *wstr); int mvgetn_wstr(int y, int x, wint_t *wstr, int n); int mvget_wstr(int y, int x, wint_t *wstr); int mvwgetn_wstr(WINDOW *win, int y, int x, wint_t *wstr, int n); int mvwget_wstr(WINDOW *win, int y, int x, wint_t *wstr); int wgetn_wstr(WINDOW *win, wint_t *wstr, int n); int wget_wstr(WINDOW *win, wint_t *wstr);</pre> |
| DESCRIPTION | <p>The <code>get_wstr()</code> and <code>wget_wstr()</code> functions get a wide character string from the terminal associated with the window <code>stdscr</code> or window <code>win</code>, respectively. The <code>mvget_wstr()</code> and <code>mvwget_wstr()</code> functions move the cursor to the position specified in <code>stdscr</code> or <code>win</code>, respectively, then get a wide character string.</p> <p>These functions call <code>wget_wch(3XCURSES)</code> and place each received character in <code>wstr</code> until a newline character, end-of-line character, or end-of-file character is received, which is also placed in <code>wstr</code>. The erase and kill characters set by the user are processed.</p> <p>The <code>getn_wstr()</code>, <code>mvgetn_wstr()</code>, <code>mvwgetn_wstr()</code> and <code>wgetn_wstr()</code> functions read at most <code>n</code> characters. These functions are used to prevent overflowing the input buffer.</p> |
| PARAMETERS | <p><i>wstr</i> Is a pointer to the area where the character string is to be placed.</p> <p><i>n</i> Is the maximum number of characters to read from input.</p> <p><i>y</i> Is the y (row) coordinate of starting position of character string to be read.</p> <p><i>x</i> Is the x (column) coordinate of starting position of character string to be read.</p> <p><i>win</i> points to the window associated with the terminal from which the character is to be read.</p> |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. |
| ERRORS | None. |

mvwget_wstr(3XCURSES)

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | `get_wch(3XCURSES)`, `getnstr(3XCURSES)`, `libcurses(3XCURSES)`,
`attributes(5)`, `standards(5)`

| | |
|----------------------|--|
| NAME | hline, mvhline, mvvline, mvwhline, mvwvline, vline, whline, wvline – use single-byte characters (and renditions) to draw lines |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int hline(chtype ch, int n); int mvhline(int y, int x, chtype ch, int n); int mvvline(int y, int x, chtype ch, int n); int mvwhline(WINDOW *win, int y, int x, chtype ch, int n); int mvwvline(WINDOW *win, int y, int x, chtype ch, int n); int vline(chtype ch, int n); int whline(WINDOW *win, chtype ch, int n); int wvline(WINDOW *win, chtype ch, int n);</pre> |
| DESCRIPTION | <p>The <code>hline()</code>, <code>vline()</code>, <code>whline()</code>, <code>wvline()</code> functions draw a horizontal or vertical line, in either the window <code>stdscr</code> or <code>win</code> starting at the current cursor position. The line is drawn using the character <code>ch</code> and is a maximum of <code>n</code> positions long, or as many as will fit into the window. If <code>ch</code> is 0 (zero), the default horizontal or vertical character is used.</p> <p>The <code>mvhline()</code>, <code>mvvline()</code>, <code>mvwhline()</code>, <code>mvwvline()</code> functions are similar to the previous group of functions but the line begins at cursor position specified by <code>x</code> and <code>y</code>.</p> <p>The functions with names ending with <code>hline()</code> draw horizontal lines proceeding towards the last column of the same line. The functions with names ending with <code>vline()</code> draw vertical lines proceeding towards the last column of the same line.</p> <p>These functions do not change the position of the cursor.</p> |
| PARAMETERS | <p><code>ch</code> Is the character used to draw the line.</p> <p><code>n</code> Is the maximum number of characters in the line.</p> <p><code>y</code> Is the y (row) coordinate for the start of the line.</p> <p><code>x</code> Is the x (column) coordinate for the start of the line.</p> <p><code>win</code> Is a pointer to a window.</p> |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. |
| ERRORS | None |

mvwhline(3XCURSES)

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | `border(3XCURSES)`, `border_set(3XCURSES)`, `hline_set(3XCURSES)`,
`libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|----------------------|---|
| NAME | hline_set, mvhline_set, mvvline_set, mvwhline_set, mvvwline_set, vline_set, whline_set, wvline_set – use complex characters (and renditions) to draw lines |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int hline_set(const cchar_t *ch, int n); int mvhline_set(int y, int x, const cchar_t *wch, int n); int mvvline_set(int y, int x, const cchar_t *wch, int n); int mvwhline_set(WINDOW *win, int y, int x, const cchar_t *wch, int n); int mvvwline_set(WINDOW *win, int y, int x, const cchar_t *wch, int n); int vline_set(const cchar_t *wch, int n); int whline_set(WINDOW *win, const cchar_t *wch, int n); int wvline_set(WINDOW *win, const cchar_t *wch, int n);</pre> |
| DESCRIPTION | <p>The <code>hline_set()</code>, <code>vline_set()</code>, <code>whline_set()</code>, <code>wvline_set()</code> functions draw a line, in either the window <code>stdscr</code> or <code>win</code> starting at the current cursor position. The line is drawn using the character <code>wch</code> and is a maximum of <code>n</code> positions long, or as many as will fit into the window. If <code>wch</code> is a null pointer, the default horizontal or vertical character is used.</p> <p>The <code>mvhline_set()</code>, <code>mvvline_set()</code>, <code>mvwhline_set()</code>, <code>mvvwline_set()</code> functions are similar to the previous group of functions but the line begins at cursor position specified by <code>x</code> and <code>y</code>.</p> <p>The functions with names ending with <code>hline_set()</code> draw horizontal lines proceeding towards the last column of the same line. The functions with names ending with <code>vline_set()</code> draw vertical lines proceeding towards the last column of the same line.</p> <p>These functions do not change the position of the cursor.</p> |
| PARAMETERS | <p><code>wch</code> Is the complex character used to draw the line.</p> <p><code>n</code> Is the maximum number of characters in the line.</p> <p><code>y</code> Is the y (row) coordinate for the start of the line.</p> <p><code>x</code> Is the x (column) coordinate for the start of the line.</p> <p><code>win</code> Is a pointer to a window.</p> |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. |

mvwhline_set(3XCURSES)

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `border(3XCURSES)`, `border_set(3XCURSES)`, `hline(3XCURSES)`,
`libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | curs_window, newwin, delwin, mvwin, subwin, derwin, mvderwin, dupwin, wsyncup, syncok, wcursyncup, wsyncdown – create curses windows |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> WINDOW *newwin(int nlines, int ncols, int begin_y, int begin_x); int delwin(WINDOW *win); int mvwin(WINDOW *win, int y, int x); WINDOW *subwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); WINDOW *derwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); int mvderwin(WINDOW *win, int par_y, int par_x); WINDOW *dupwin(WINDOW *win); void wsyncup(WINDOW *win); int syncok(WINDOW *win, bool bf); void wcursyncup(WINDOW *win); void wsyncdown(WINDOW *win);</pre> |
| DESCRIPTION | <p>The <code>newwin()</code> routine creates and returns a pointer to a new window with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. The upper left-hand corner of the window is at line <i>begin_y</i>, column <i>begin_x</i>. If either <i>nlines</i> or <i>ncols</i> is zero, they default to LINES — <i>begin_y</i> and COLS — <i>begin_x</i>. A new full-screen window is created by calling <code>newwin(0, 0, 0, 0)</code>.</p> <p>The <code>delwin()</code> routine deletes the named window, freeing all memory associated with it. Subwindows must be deleted before the main window can be deleted.</p> <p>The <code>mvwin()</code> routine moves the window so that the upper left-hand corner is at position (<i>x</i>, <i>y</i>). If the move would cause the window to be off the screen, it is an error and the window is not moved. Moving subwindows is allowed, but should be avoided.</p> <p>The <code>subwin()</code> routine creates and returns a pointer to a new window with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. The window is at position (<i>begin_y</i>, <i>begin_x</i>) on the screen. (This position is relative to the screen, and not to the window <i>orig</i>.) The window is made in the middle of the window <i>orig</i>, so that changes made to one window will affect both windows. The subwindow shares memory with the window <i>orig</i>. When using this routine, it is necessary to call <code>touchwin()</code> or <code>touchline()</code> on <i>orig</i> before calling <code>wrefresh()</code> on the subwindow.</p> <p>The <code>derwin()</code> routine is the same as <code>subwin()</code>, except that <i>begin_y</i> and <i>begin_x</i> are relative to the origin of the window <i>orig</i> rather than the screen. There is no difference between the subwindows and the derived windows.</p> |

mvwin(3CURSES)

The `mvderwin()` routine moves a derived window (or subwindow) inside its parent window. The screen-relative parameters of the window are not changed. This routine is used to display different parts of the parent window at the same physical position on the screen.

The `dupwin()` routine creates an exact duplicate of the window *win*.

Each `curses` window maintains two data structures: the character image structure and the status structure. The character image structure is shared among all windows in the window hierarchy (that is, the window with all subwindows). The status structure, which contains information about individual line changes in the window, is private to each window. The routine `wrefresh()` uses the status data structure when performing screen updating. Since status structures are not shared, changes made to one window in the hierarchy may not be properly reflected on the screen.

The routine `wsyncup()` causes the changes in the status structure of a window to be reflected in the status structures of its ancestors. If `syncok()` is called with second argument `TRUE` then `wsyncup()` is called automatically whenever there is a change in the window.

The routine `wcursyncup()` updates the current cursor position of all the ancestors of the window to reflect the current cursor position of the window.

The routine `wsyncdown()` updates the status structure of the window to reflect the changes in the status structures of its ancestors. Applications seldom call this routine because it is called automatically by `wrefresh()`.

RETURN VALUES

Routines that return an integer return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`delwin()` returns the integer `ERR` upon failure and `OK` upon successful completion.

Routines that return pointers return `NULL` on error.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_refresh(3CURSES)`, `curs_touch(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

If many small changes are made to the window, the `wsyncup()` option could degrade performance.

`mvwin(3CURSES)`

Note that `syncok()` may be a macro.

mvwin(3XCURSES)

| NAME | mvwin – move window | | | | | | |
|----------------------|---|----------------|-------------------------------------|---------------------|---|----------|--|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int mvwin(WINDOW *win, int y, int x);</pre> | | | | | | |
| PARAMETERS | <table><tr><td><i>win</i></td><td>Is a pointer to the window to move.</td></tr><tr><td><i>y</i></td><td>Is the y (row) coordinate of the upper left corner of the window.</td></tr><tr><td><i>x</i></td><td>Is the x (column) coordinate of the upper left corner of the window.</td></tr></table> | <i>win</i> | Is a pointer to the window to move. | <i>y</i> | Is the y (row) coordinate of the upper left corner of the window. | <i>x</i> | Is the x (column) coordinate of the upper left corner of the window. |
| <i>win</i> | Is a pointer to the window to move. | | | | | | |
| <i>y</i> | Is the y (row) coordinate of the upper left corner of the window. | | | | | | |
| <i>x</i> | Is the x (column) coordinate of the upper left corner of the window. | | | | | | |
| DESCRIPTION | The mvwin() function moves the specified window (or subwindow), placing its upper left corner at the positions specified by <i>x</i> and <i>y</i> . The entire window must fit within the physical boundaries of the screen or an error results. In the case of a subwindow, the window must remain within the boundaries of the parent window. | | | | | | |
| RETURN VALUES | On success, the mvwin() function returns OK. Otherwise, it returns ERR. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Standard</td></tr><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | derwin(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5) | | | | | | |

| NAME | curs_inch, inch, winch, mvinch, mvwinch – get a character and its attributes from a curses window | | | | | | |
|--------------------|--|----------------|-------------------------------|--------------|--------------------------------|---------|--|
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> ctype inch(void); ctype winch(WINDOW *<i>win</i>); ctype mvinch(int <i>y</i>, int <i>x</i>); ctype mvwinch(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>);</pre> | | | | | | |
| DESCRIPTION | With these routines, the character, of type <code>ctype</code> , at the current position in the named window is returned. If any attributes are set for that position, their values are OR-ed into the value returned. Constants defined in <code><curses.h></code> can be used with the logical AND (<code>&</code>) operator to extract the character or attributes alone. | | | | | | |
| Attributes | <p>The following bit-masks can be AND-ed with characters returned by <code>winch()</code>.</p> <table> <tr> <td>A_CHARTEXT</td> <td>Bit-mask to extract character</td> </tr> <tr> <td>A_ATTRIBUTES</td> <td>Bit-mask to extract attributes</td> </tr> <tr> <td>A_COLOR</td> <td>Bit-mask to extract color-pair field information</td> </tr> </table> | A_CHARTEXT | Bit-mask to extract character | A_ATTRIBUTES | Bit-mask to extract attributes | A_COLOR | Bit-mask to extract color-pair field information |
| A_CHARTEXT | Bit-mask to extract character | | | | | | |
| A_ATTRIBUTES | Bit-mask to extract attributes | | | | | | |
| A_COLOR | Bit-mask to extract color-pair field information | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that all of these routines may be macros.</p> | | | | | | |

mvwinch(3XCURSES)

| NAME | inch, mvinch, mvwinch, winch – return a single-byte character (with rendition) | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> chtype inch(void); chtype mvinch(int <i>y</i>, int <i>x</i>); chtype mvwinch(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>); chtype winch(WINDOW *<i>win</i>);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>inch()</code> and <code>winch()</code> functions return the <code>chtype</code> character located at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively. The <code>mvinch()</code> and <code>mvwinch()</code> functions return the <code>chtype</code> character located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The complete character/attribute pair will be returned. The character or attributes can be extracted by performing a bitwise AND on the returned value, using the constants <code>A_CHARTEXT</code>, <code>A_ATTRIBUTES</code>, and <code>A_COLOR</code>.</p> | | | | | | |
| PARAMETERS | <p><i>y</i> Is the <i>y</i> (row) coordinate of the position of the character to be returned.</p> <p><i>x</i> Is the <i>x</i> (column) coordinate of the position of the character to be returned.</p> <p><i>win</i> Is a pointer to the window that contains the character to be returned.</p> | | | | | | |
| RETURN VALUES | On success, these functions return the specified character and rendition. Otherwise, they return <code>ERR</code> . | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Standard</td></tr><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>addch(3XCURSES)</code> , <code>attroff(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

| NAME | curs_inchstr, inchstr, inchnstr, winchstr, winchnstr, mvinchstr, mvwinchnstr, mvwinchstr, mvwinchnstr – get a string of characters (and attributes) from a curses window | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int inchstr(chtype *chstr); int inchnstr(chtype *chstr, int n); int winchstr(WINDOW *win, chtype *chstr); int winchnstr(WINDOW *win, chtype *chstr, int n); int mvinchstr(int y, int x, chtype *chstr); int mvwinchnstr(int y, int x, chtype *chstr, int n); int mvwinchstr(WINDOW *win, int y, int x, chtype *chstr); int mvwinchnstr(WINDOW *win, int y, int x, chtype *chstr, int n);</pre> | | | | |
| DESCRIPTION | With these routines, a string of type chtype, starting at the current cursor position in the named window and ending at the right margin of the window, is returned. The four functions with <i>n</i> as the last argument, return the string at most <i>n</i> characters long. Constants defined in <curses.h> can be used with the & (logical AND) operator to extract the character or the attribute alone from any position in the <i>chstr</i> (see curs_inch(3CURSES)). | | | | |
| RETURN VALUES | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curs_inch(3CURSES), curses(3CURSES), attributes(5) | | | | |
| NOTES | <p>The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.</p> <p>Note that all routines except winchnstr() may be macros.</p> | | | | |

mvwinchnstr(3XCURSES)

| | | | | | | | | | |
|--------------------|---|--------------|---|----------|--|----------|--|----------|---|
| NAME | <code>inchstr</code> , <code>inchstr</code> , <code>mvinchnstr</code> , <code>mvinchstr</code> , <code>mvwinchnstr</code> , <code>mvwinchstr</code> , <code>winchnstr</code> , <code>winchstr</code> – retrieve a single-byte character string (with rendition) | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int inchnstr(chtype *chstr, int n); int inchstr(chtype *chstr); int mvinchnstr(int y, int x, chtype *chstr, int n); int mvinchstr(int y, int x, chtype *chstr); int mvwinchnstr(WINDOW *win, int y, int x, chtype *chstr, int n); int mvwinchstr(WINDOW *win, int y, int x, chtype *chstr); int winchnstr(WINDOW *win, chtype *chstr, int n); int winchstr(WINDOW *win, chtype *chstr);</pre> | | | | | | | | |
| DESCRIPTION | <p>The <code>inchstr()</code> and <code>winchstr()</code> functions retrieve the character string (with rendition) starting at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively, and ending at the right margin. The <code>mvinchstr()</code> and <code>mvwinchstr()</code> functions retrieve the character string located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>inchnstr()</code>, <code>winchnstr()</code>, <code>mvinchnstr()</code>, and <code>mvwinchnstr()</code> functions retrieve at most <i>n</i> characters from the window <code>stdscr</code> and <i>win</i>, respectively. The former two functions retrieve the string, starting at the current cursor position; the latter two commands retrieve the string, starting at the position specified by the <i>x</i> and <i>y</i> parameters.</p> <p>All these functions store the retrieved character string in the object pointed to by <i>chstr</i>.</p> <p>The complete character/attribute pair is retrieved. The character or attributes can be extracted by performing a bitwise AND on the retrieved value, using the constants <code>A_CHARTEXT</code>, <code>A_ATTRIBUTES</code>, and <code>A_COLOR</code>. The character string can also be retrieved without attributes by using <code>instr(3XCURSES)</code> set of functions.</p> | | | | | | | | |
| PARAMETERS | <table><tr><td><i>chstr</i></td><td>Is a pointer to an object that can hold the retrieved character string.</td></tr><tr><td><i>n</i></td><td>Is the number of characters not to exceed when retrieving <i>chstr</i>.</td></tr><tr><td><i>y</i></td><td>Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved.</td></tr><tr><td><i>x</i></td><td>Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved.</td></tr></table> | <i>chstr</i> | Is a pointer to an object that can hold the retrieved character string. | <i>n</i> | Is the number of characters not to exceed when retrieving <i>chstr</i> . | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. |
| <i>chstr</i> | Is a pointer to an object that can hold the retrieved character string. | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when retrieving <i>chstr</i> . | | | | | | | | |
| <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | | | | | | | | |
| <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | | | | | | | | |

`mvwinchnstr(3XCURSES)`

win Is a pointer to the window in which the string is to be retrieved.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `inch(3XCURSES)`, `innstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

mvwinchstr(3CURSES)

| NAME | <code> curs_ inchstr, inchstr, inchnstr, winchstr, winchnstr, mvinchstr, mvinchnstr, mvwinchstr, mvwinchnstr</code> – get a string of characters (and attributes) from a curses window | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int inchstr(chtype *chstr); int inchnstr(chtype *chstr, int n); int winchstr(WINDOW *win, chtype *chstr); int winchnstr(WINDOW *win, chtype *chstr, int n); int mvinchstr(int y, int x, chtype *chstr); int mvinchnstr(int y, int x, chtype *chstr, int n); int mvwinchstr(WINDOW *win, int y, int x, chtype *chstr); int mvwinchnstr(WINDOW *win, int y, int x, chtype *chstr, int n);</pre> | | | | |
| DESCRIPTION | With these routines, a string of type <code>chtype</code> , starting at the current cursor position in the named window and ending at the right margin of the window, is returned. The four functions with <code>n</code> as the last argument, return the string at most <code>n</code> characters long. Constants defined in <code><curses.h></code> can be used with the <code>&</code> (logical AND) operator to extract the character or the attribute alone from any position in the <code>chstr</code> (see <code>curs_ inch(3CURSES)</code>). | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_ inch(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . Note that all routines except <code>winchnstr()</code> may be macros. | | | | |

| | |
|--------------------|--|
| NAME | inchstr, inchstr, mvinchnstr, mvinchstr, mvwinchnstr, mvwinchstr, winchnstr, winchstr – retrieve a single-byte character string (with rendition) |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int inchnstr(chtype *chstr, int n); int inchstr(chtype *chstr); int mvinchnstr(int y, int x, chtype *chstr, int n); int mvinchstr(int y, int x, chtype *chstr); int mvwinchnstr(WINDOW *win, int y, int x, chtype *chstr, int n); int mvwinchstr(WINDOW *win, int y, int x, chtype *chstr); int winchnstr(WINDOW *win, chtype *chstr, int n); int winchstr(WINDOW *win, chtype *chstr);</pre> |
| DESCRIPTION | <p>The <code>inchstr()</code> and <code>winchstr()</code> functions retrieve the character string (with rendition) starting at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively, and ending at the right margin. The <code>mvinchnstr()</code> and <code>mvwinchstr()</code> functions retrieve the character string located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>inchnstr()</code>, <code>winchnstr()</code>, <code>mvinchnstr()</code>, and <code>mvwinchnstr()</code> functions retrieve at most <i>n</i> characters from the window <code>stdscr</code> and <i>win</i>, respectively. The former two functions retrieve the string, starting at the current cursor position; the latter two commands retrieve the string, starting at the position specified by the <i>x</i> and <i>y</i> parameters.</p> <p>All these functions store the retrieved character string in the object pointed to by <i>chstr</i>.</p> <p>The complete character/attribute pair is retrieved. The character or attributes can be extracted by performing a bitwise AND on the retrieved value, using the constants <code>A_CHARTEXT</code>, <code>A_ATTRIBUTES</code>, and <code>A_COLOR</code>. The character string can also be retrieved without attributes by using <code>instr(3XCURSES)</code> set of functions.</p> |
| PARAMETERS | <p><i>chstr</i> Is a pointer to an object that can hold the retrieved character string.</p> <p><i>n</i> Is the number of characters not to exceed when retrieving <i>chstr</i>.</p> <p><i>y</i> Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved.</p> <p><i>x</i> Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved.</p> |

mvwinchstr(3XCURSES)

win Is a pointer to the window in which the string is to be retrieved.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `inch(3XCURSES)`, `innstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_instr, instr, innstr, winstr, winnstr, mvinstr, mvinnstr, mvwinstr, mvwinstr – get a string of characters from a curses window | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int instr(char *str); int innstr(char *str, int n); int winstr(WINDOW *win, char *str); int winnstr(WINDOW *win, char *str, int n); int mvinstr(int y, int x, char *str); int mvinnstr(int y, int x, char *str, int n); int mvwinstr(WINDOW *win, int y, int x, char *str); int mvwinstr(WINDOW *win, int y, int x, char *str, int n);</pre> | | | | |
| DESCRIPTION | These routines return a string of characters in <i>str</i> , starting at the current cursor position in the named window and ending at the right margin of the window. Attributes are stripped from the characters. The four functions with <i>n</i> as the last argument return the string at most <i>n</i> characters long. | | | | |
| RETURN VALUES | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), attributes(5) | | | | |
| NOTES | <p>The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.</p> <p>Note that all routines except winnstr() may be macros.</p> | | | | |

mvwinstr(3XCURSES)

| | | | | | | | | | | | |
|--------------------|---|------------|---|----------|--|----------|--|----------|---|------------|--|
| NAME | <code>innstr</code> , <code>instr</code> , <code>mvinnstr</code> , <code>mvinstr</code> , <code>mvwinstr</code> , <code>mvwinstr</code> , <code>winnstr</code> , <code>winstr</code> – retrieve a multibyte character string (without rendition) | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int innstr(char *str, int n); int instr(char *str); int mvinnstr(int y, int x, char *str, int n); int mvinstr(int y, int x, char *str); int mvwinstr(WINDOW *win, int y, int x, char *str, int n); int mvwinstr(WINDOW *win, int y, int x, char *str); int winstr(WINDOW *win, char *str); int winstr(WINDOW *win, char *str, int n);</pre> | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>str</i></td><td>Is a pointer to an object that can hold the retrieved multibyte character string.</td></tr><tr><td><i>n</i></td><td>Is the number of characters not to exceed when retrieving <i>str</i>.</td></tr><tr><td><i>y</i></td><td>Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved.</td></tr><tr><td><i>x</i></td><td>Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved.</td></tr><tr><td><i>win</i></td><td>Is a pointer to the window in which the string is to be retrieved.</td></tr></table> | <i>str</i> | Is a pointer to an object that can hold the retrieved multibyte character string. | <i>n</i> | Is the number of characters not to exceed when retrieving <i>str</i> . | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | <i>win</i> | Is a pointer to the window in which the string is to be retrieved. |
| <i>str</i> | Is a pointer to an object that can hold the retrieved multibyte character string. | | | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when retrieving <i>str</i> . | | | | | | | | | | |
| <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | | | | | | | | | | |
| <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the string is to be retrieved. | | | | | | | | | | |
| DESCRIPTION | <p>The <code>instr()</code> and <code>winstr()</code> functions retrieve a multibyte character string (without attributes) starting at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively, and ending at the right margin. The <code>mvinstr()</code> and <code>mvwinstr()</code> functions retrieve a multibyte character string located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>innstr()</code>, <code>winnstr()</code>, <code>mvinnstr()</code>, and <code>mvwinstr()</code> functions retrieve at most <i>n</i> characters from the window <code>stdscr</code> and <i>win</i>, respectively. The former two functions retrieve the string starting at the current cursor position; the latter two commands return the string, starting at the position specified by the <i>x</i> and <i>y</i> parameters.</p> | | | | | | | | | | |

`mvwinnstr(3XCURSES)`

All these functions store the retrieved string in the object pointed to by *str*. They only store complete multibyte characters. If the area pointed to by *str* is not large enough to hold at least one character, these functions fail.

Only the character portion of the character/rendition pair is returned. To return the complete character/rendition pair, use `winchstr()`.

ERRORS OK Successful completion.

ERR An error occurred.

USAGE All functions except `winnstr()` may be macros.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `inch(3XCURSES)`, `inchstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

mvwinwstr(3CURSES)

| NAME | <code> curs_inwstr, inwstr, innwstr, winwstr, winnwstr, mvinwstr, mvinnwstr, mvwinwstr, mvwinwstr</code> – get a string of <code>wchar_t</code> characters from a curses window | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses[library ..] #include <curses.h> int inwstr(wchar_t *wstr); int innwstr(wchar_t *wstr, int n); int winwstr(WINDOW *win, wchar_t *wstr); int winnwstr(WINDOW *win, wchar_t *wstr, int n); int mvinwstr(int y, int x, wchar_t *wstr); int mvinnwstr(int y, int x, wchar_t *wstr, int n); int mvwinwstr(WINDOW *win, int y, int x, wchar_t *wstr); int mvwinnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);</pre> | | | | |
| DESCRIPTION | These routines return the string of <code>wchar_t</code> characters in <code>wstr</code> starting at the current cursor position in the named window and ending at the right margin of the window. Attributes are stripped from the characters. The four functions with <code>n</code> as the last argument return the string at most <code>n</code> <code>wchar_t</code> characters long. | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header file <code><curses.h></code> automatically includes the header files <code><stdio.h></code> , <code><unctrl.h></code> and <code><wctype.h></code> . Note that all routines except <code>winnwstr()</code> may be macros. | | | | |

| | |
|--------------------|---|
| NAME | innwstr, inwstr, mvinnwstr, mvinwstr, mvwinnwstr, mvwinwstr, winnwstr, winwstr – retrieve a wide character string (without rendition) |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int innwstr(wchar_t *wstr, int n); int inwstr(wchar_t *wstr); int mvinnwstr(int y, int x, wchar_t *wstr, int n); int mvinwstr(int y, int x, wchar_t *wstr); int mvwinnwstr(WINDOW*win, int y, int x, wchar_t *wstr, int n); int mvwinwstr(WINDOW*win, int y, int x, wchar_t *wstr); int winnwstr(WINDOW*win, wchar_t *wstr); int winwstr(WINDOW*win, wchar_t *wstr); int winnwstr(WINDOW*win, wchar_t *wstr, int n);</pre> |
| PARAMETERS | <p><i>wstr</i> Is a pointer to an object that can hold the retrieved multibyte character string.</p> <p><i>n</i> Is the number of characters not to exceed when retrieving <i>wstr</i>.</p> <p><i>y</i> Is the y (row) coordinate of the starting position of the string to be retrieved.</p> <p><i>x</i> Is the x (column) coordinate of the starting position of the string to be retrieved.</p> <p><i>win</i> Is a pointer to the window in which the string is to be retrieved.</p> |
| DESCRIPTION | <p>The <code>inwstr()</code> and <code>winwstr()</code> functions retrieve a wide character string (without attributes) starting at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively, and ending at the right margin. The <code>mvinwstr()</code> and <code>mvwinwstr()</code> functions retrieve a wide character string located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>innwstr()</code>, <code>winnwstr()</code>, <code>mvinnwstr()</code>, and <code>mvwinnwstr()</code> functions retrieve at most <i>n</i> characters from the window <code>stdscr</code> and <i>win</i>, respectively. The former two functions retrieve the string starting at the current cursor position; the latter two commands return the string, starting at the position specified by the <i>x</i> and <i>y</i> parameters.</p> <p>All these functions store the retrieved string in the object pointed to by <i>wstr</i>. They only store complete wide characters. If the area pointed to by <i>wstr</i> is not large enough to hold at least one character, these functions fail.</p> |

mvwinnwstr(3XCURSES)

RETURN VALUES Only the character portion of the character/rendition pair is returned. To return the complete character/rendition pair, use `win_wchstr(3XCURSES)`.
On success, the `innwstr()`, `mvinwstr()`, `mvwinnwstr()`, and `winnwstr()` functions return OK. Otherwise, they return ERR.
On success, the `innwstr()`, `mvinwstr()`, `mvwinnwstr()`, and `winnwstr()` functions return the number of characters read into the string. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `in_wch(3XCURSES)`, `in_wchnstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|----------------------|---|
| NAME | curs_ <i>insch</i> , <i>insch</i> , <i>winsch</i> , <i>mvinsch</i> , <i>mvwinsch</i> – insert a character before the character under the cursor in a curses window |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i>... -lcurses [<i>library</i> ...] #include <curses.h> int insch(chtype <i>ch</i>); int winsch(WINDOW *<i>win</i>, chtype <i>ch</i>); int mvinsch(int <i>y</i>, int <i>x</i>, chtype <i>ch</i>); int mvwinsch(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>, chtype <i>ch</i>);</pre> |
| DESCRIPTION | With these routines, the character <i>ch</i> is inserted before the character under the cursor. All characters to the right of the cursor are moved one space to the right, with the possibility of the rightmost character on the line being lost. The cursor position does not change (after moving to <i>y</i> , <i>x</i> , if specified). (This does not imply use of the hardware insert character feature.) |
| RETURN VALUES | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: |

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

| | |
|-----------------|---|
| SEE ALSO | curses(3CURSES), attributes(5) |
| NOTES | <p>The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.</p> <p>Note that <i>insch()</i>, <i>mvinsch()</i>, and <i>mvwinsch()</i> may be macros.</p> |

mvwinsch(3XCURSES)

NAME | insch, winsch, mvinsch, mvwinsch – insert a character

SYNOPSIS | `cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \`
`-R /usr/xpg4/lib -lcurses [library...]`

c89 | `[flag...] file... -lcurses [library...]`

```
#include <curses.h>
int insch(chtype ch);
int mvinsch(int y, int x, chtype ch);
int mvwinsch(WINDOW *win, int y, int x, chtype ch);
int winsch(WINDOW *win, chtype ch);
```

PARAMETERS

| | |
|------------|--|
| <i>ch</i> | Is the character to be inserted. |
| <i>y</i> | Is the y (row) coordinate of the position of the character. |
| <i>x</i> | Is the x (column) coordinate of the position of the character. |
| <i>win</i> | Is a pointer to the window in which the character is to be inserted. |

DESCRIPTION | These functions insert the character and rendition from *ch* into the current or specified window at the current or specified position.

These functions do not perform wrapping and do not advance the cursor position. These functions perform special-character processing, with the exception that if a newline is inserted into the last line of a window and scrolling is not enabled, the behavior is unspecified.

RETURN VALUES | Upon successful completion, these functions return OK. Otherwise, they return ERR.

ERRORS | No errors are defined.

USAGE | These functions are only guaranteed to operate reliably on character sets in which each character fits into a single byte, whose attributes can be expressed using only constants with the A_ prefix.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | ins_wch(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

| NAME | curs_insstr, insstr, insnstr, winsstr, winsnstr, mvinsstr, mvinsnstr, mvwinsstr, mvwinsnstr – insert string before character under the cursor in a curses window | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int insstr(char *str); int insnstr(char *str, int n); int winsstr(WINDOW *win, char *str); int winsnstr(WINDOW *win, char *str, int n); int mvinsstr(int y, int x, char *str); int mvinsnstr(int y, int x, char *str, int n); int mvwinsstr(WINDOW *win, int y, int x, char *str); int mvwinsnstr(WINDOW *win, int y, int x, char *str, int n);</pre> | | | | |
| DESCRIPTION | <p>With these routines, a character string (as many characters as will fit on the line) is inserted before the character under the cursor. All characters to the right of the cursor are moved to the right, with the possibility of the rightmost characters on the line being lost. The cursor position does not change (after moving to <i>y, x</i>, if specified). (This does not imply use of the hardware insert character feature.) The four routines with <i>n</i> as the last argument insert at most <i>n</i> characters. If <i>n</i> ≤ 0, then the entire string is inserted.</p> <p>If a character in <i>str</i> is a tab, newline, carriage return or backspace, the cursor is moved appropriately within the window. A newline also does a <code>clrtoeol()</code> before moving. Tabs are considered to be at every eighth column. If a character in <i>str</i> is another control character, it is drawn in the <code>^X</code> notation. Calling <code>winch()</code> after adding a control character (and moving to it, if necessary) does not return the control character, but instead returns the representation of the control character.</p> | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_clear(3CURSES)</code> , <code>curs_inch(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . | | | | |

`mvwinsnstr(3CURSES)`

Note that all but `winsnstr()` may be macros.

| | | | | | | | | | | | |
|--------------------|---|------------|--|----------|--|----------|--|----------|---|------------|---|
| NAME | insnstr, insstr, mvinsnstr, mvinsstr, mvwinsnstr, mvwinsstr, winsnstr, winsstr – insert a multibyte character string | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int insnstr(const char *str, int n); int insstr(const char *str); int mvinsnstr(int y, int x, const char *str, int n); int mvinsstr(int y, int x, const char *str); int mvwinsnstr(WINDOW *win, int y, int x, const char *str, int n); int mvwinsstr(WINDOW *win, int y, int x, const char *str); int winsnstr(WINDOW *win, const char *str, int n); int winsstr(WINDOW *win, const char *str);</pre> | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>str</i></td> <td>Is a pointer to the string to be inserted.</td> </tr> <tr> <td><i>n</i></td> <td>Is the number of characters not to exceed when inserting <i>str</i>. If <i>n</i> is less than 1, the entire string is inserted.</td> </tr> <tr> <td><i>y</i></td> <td>Is the <i>y</i> (row) coordinate of the starting position of the string.</td> </tr> <tr> <td><i>x</i></td> <td>Is the <i>x</i> (column) coordinate of the starting position of the string.</td> </tr> <tr> <td><i>win</i></td> <td>Is a pointer to the window in which the string is to be inserted.</td> </tr> </table> | <i>str</i> | Is a pointer to the string to be inserted. | <i>n</i> | Is the number of characters not to exceed when inserting <i>str</i> . If <i>n</i> is less than 1, the entire string is inserted. | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string. | <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string. | <i>win</i> | Is a pointer to the window in which the string is to be inserted. |
| <i>str</i> | Is a pointer to the string to be inserted. | | | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when inserting <i>str</i> . If <i>n</i> is less than 1, the entire string is inserted. | | | | | | | | | | |
| <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string. | | | | | | | | | | |
| <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the string is to be inserted. | | | | | | | | | | |
| DESCRIPTION | <p>The <code>insstr()</code> function inserts <i>str</i> at the current cursor position of the <code>stdscr</code> window. The <code>winsstr()</code> function performs the identical action, but in window <i>win</i>. The <code>mvinsstr()</code> and <code>mvwinsstr()</code> functions insert the character string at the starting position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <i>win</i>).</p> <p>The <code>insnstr()</code>, <code>winsnstr()</code>, <code>mvinsnstr()</code>, and <code>mvwinsnstr()</code> functions insert <i>n</i> characters to the window or as many as will fit on the line. If <i>n</i> is less than 1, the entire string is inserted or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <i>x</i> and <i>y</i> parameters.</p> <p>All characters to the right of inserted characters are moved to the right. Characters that don't fit on the current line are discarded. The cursor is left at the point of insertion.</p> | | | | | | | | | | |

mvwinsnstr(3XCURSES)

If a character in *str* is a newline, carriage return, backspace, or tab, the cursor is moved appropriately. The cursor is moved to the next tab stop for each tab character (by default, tabs are eight characters apart). If the character is a control character other than those previously mentioned, the character is inserted using $\^x$ notation, where *x* is a printable character. `clrtoeol(3XCURSES)` is automatically done before a newline.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addchstr(3XCURSES)`, `addstr(3XCURSES)`, `clrtoeol(3XCURSES)`, `ins_nwstr(3XCURSES)`, `insch(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_inswstr, inswstr, insnwstr, winswstr, winsnwstr, mvinswstr, mvinsnwstr, mvwinswstr, mvwinsnwstr – insert wchar_t string before character under the cursor in a curses window | | | | |
|---------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int inswstr(wchar_t *wstr); int insnwstr(wchar_t *wstr, int n); int winswstr(WINDOW *win, wchar_t *wstr); int winsnwstr(WINDOW *win, wchar_t *wstr, int n); int mvinswstr(int y, int x, wchar_t *wstr); int mvinsnwstr(int y, int x, wchar_t *wstr, int n); int mvwinswstr(WINDOW *win, int y, int x, wchar_t *wstr); int mvwinsnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);</pre> | | | | |
| DESCRIPTION | <p>These routines insert a wchar_t character string (as many wchar_t characters as will fit on the line) before the character under the cursor. All characters to the right of the cursor are moved to the right, with the possibility of the rightmost characters on the line being lost. The cursor position does not change (after moving to <i>y, x</i>, if specified). (This does not imply use of the hardware insert character feature.) The four routines with <i>n</i> as the last argument insert at most <i>n</i> wchar_t characters. If <i>n</i> ≤ 0, then the entire string is inserted.</p> <p>If a character in <i>wstr</i> is a tab, newline, carriage return, or backspace, the cursor is moved appropriately within the window. A newline also does a clrtoeol(3CURSES) before moving. Tabs are considered to be at every eighth column. If a character in <i>wstr</i> is another control character, it is drawn in the ^X notation. Calling winwch(3CURSES) after adding a control character (and moving to it, if necessary) does not return the control character, but instead returns the representation of the control character.</p> | | | | |
| RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | clrtoeol(3CURSES), curses(3CURSES), winwch(3CURSES), attributes(5) | | | | |
| NOTES | The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <wchar.h>. | | | | |

`mvwinsnwstr(3CURSES)`

Note that all but `winsnwstr()` may be macros.

| | | | | | | | | | | | |
|--------------------|---|-------------|--|----------|---|----------|---|----------|--|------------|---|
| NAME | ins_nwstr, ins_wstr, mvins_nwstr, mvins_wstr, mvwins_nwstr, mvwins_wstr, wins_nwstr, wins_wstr – insert a wide character string | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int ins_nwstr(const wchar_t *wstr, int n); int ins_wstr(const wchar_t *wstr); int mvins_nwstr(int y, int x, const wchar_t *wstr, int n); int mvins_wstr(int y, int x, const wchar_t *wstr); int mvwins_nwstr(WINDOW *win, int y, int x, const wchar_t *wstr, int n); int mvwins_wstr(WINDOW *win, int y, int x, const wchar_t *wstr); int wins_nwstr(WINDOW *win, const wchar_t *wstr, int n); int wins_wstr(WINDOW *win, const wchar_t *wstr);</pre> | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>wstr</i></td> <td>Is a pointer to the string to be inserted.</td> </tr> <tr> <td><i>n</i></td> <td>Is the number of characters not to exceed when inserting <i>wstr</i>. If <i>n</i> is less than 1, the entire string is inserted.</td> </tr> <tr> <td><i>y</i></td> <td>Is the y (row) coordinate of the starting position of the string.</td> </tr> <tr> <td><i>x</i></td> <td>Is the x (column) coordinate of the starting position of the string.</td> </tr> <tr> <td><i>win</i></td> <td>Is a pointer to the window in which the string is to be inserted.</td> </tr> </table> | <i>wstr</i> | Is a pointer to the string to be inserted. | <i>n</i> | Is the number of characters not to exceed when inserting <i>wstr</i> . If <i>n</i> is less than 1, the entire string is inserted. | <i>y</i> | Is the y (row) coordinate of the starting position of the string. | <i>x</i> | Is the x (column) coordinate of the starting position of the string. | <i>win</i> | Is a pointer to the window in which the string is to be inserted. |
| <i>wstr</i> | Is a pointer to the string to be inserted. | | | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when inserting <i>wstr</i> . If <i>n</i> is less than 1, the entire string is inserted. | | | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of the starting position of the string. | | | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of the starting position of the string. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the string is to be inserted. | | | | | | | | | | |
| DESCRIPTION | <p>The <code>ins_wstr()</code> function inserts <i>wstr</i> at the current cursor position of the <code>stdscr</code> window. The <code>wins_wstr()</code> function performs the identical action, but in window <i>win</i>. The <code>mvins_wstr()</code> and <code>mvwins_wstr()</code> functions insert <i>wstr</i> string at the starting position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>ins_nwstr()</code>, <code>wins_nwstr()</code>, <code>mvins_nwstr()</code>, and <code>mvwins_nwstr()</code> functions insert <i>n</i> characters to the window or as many as will fit on the line. If <i>n</i> is less than 1, the entire string is inserted or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <i>x</i> and <i>y</i> parameters.</p> | | | | | | | | | | |

mvwins_nwstr(3XCURSES)

All characters to the right of inserted characters are moved to the right. Characters that don't fit on the current line are discarded. The cursor is left at the point of insertion.

If a character in *wstr* is a newline, carriage return, backspace, or tab, the cursor is moved appropriately. The cursor is moved to the next tab stop for each tab character (by default, tabs are eight characters apart). If the character is a control character other than those previously mentioned, the character is inserted using \hat{x} notation, where x is a printable character. `clrtoeol(3XCURSES)` is automatically done before a newline.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `add_wchnstr(3XCURSES)`, `addnwstr(3XCURSES)`, `clrtoeol(3XCURSES)`, `ins_wch(3XCURSES)`, `insnstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_insstr, insstr, insnstr, winsstr, winsnstr, mvinsstr, mvinsnstr, mvwinsstr, mvwinsnstr – insert string before character under the cursor in a curses window | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int insstr(char *str); int insnstr(char *str, int n); int winsstr(WINDOW *win, char *str); int winsnstr(WINDOW *win, char *str, int n); int mvinsstr(int y, int x, char *str); int mvinsnstr(int y, int x, char *str, int n); int mvwinsstr(WINDOW *win, int y, int x, char *str); int mvwinsnstr(WINDOW *win, int y, int x, char *str, int n);</pre> | | | | |
| DESCRIPTION | <p>With these routines, a character string (as many characters as will fit on the line) is inserted before the character under the cursor. All characters to the right of the cursor are moved to the right, with the possibility of the rightmost characters on the line being lost. The cursor position does not change (after moving to <i>y</i>, <i>x</i>, if specified). (This does not imply use of the hardware insert character feature.) The four routines with <i>n</i> as the last argument insert at most <i>n</i> characters. If <i>n</i> ≤ 0, then the entire string is inserted.</p> <p>If a character in <i>str</i> is a tab, newline, carriage return or backspace, the cursor is moved appropriately within the window. A newline also does a <code>clrtoeol()</code> before moving. Tabs are considered to be at every eighth column. If a character in <i>str</i> is another control character, it is drawn in the <code>^X</code> notation. Calling <code>winch()</code> after adding a control character (and moving to it, if necessary) does not return the control character, but instead returns the representation of the control character.</p> | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_clear(3CURSES)</code> , <code>curs_inch(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . | | | | |

`mvwinsstr(3CURSES)`

Note that all but `winsnstr()` may be macros.

| | | | | | | | | | | | |
|--------------------|---|------------|--|----------|--|----------|--|----------|---|------------|---|
| NAME | insnstr, insstr, mvinsnstr, mvinsstr, mvwinsnstr, mvwinsstr, winsnstr, winsstr – insert a multibyte character string | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int insnstr(const char *str, int n); int insstr(const char *str); int mvinsnstr(int y, int x, const char *str, int n); int mvinsstr(int y, int x, const char *str); int mvwinsnstr(WINDOW *win, int y, int x, const char *str, int n); int mvwinsstr(WINDOW *win, int y, int x, const char *str); int winsnstr(WINDOW *win, const char *str, int n); int winsstr(WINDOW *win, const char *str);</pre> | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>str</i></td> <td>Is a pointer to the string to be inserted.</td> </tr> <tr> <td><i>n</i></td> <td>Is the number of characters not to exceed when inserting <i>str</i>. If <i>n</i> is less than 1, the entire string is inserted.</td> </tr> <tr> <td><i>y</i></td> <td>Is the <i>y</i> (row) coordinate of the starting position of the string.</td> </tr> <tr> <td><i>x</i></td> <td>Is the <i>x</i> (column) coordinate of the starting position of the string.</td> </tr> <tr> <td><i>win</i></td> <td>Is a pointer to the window in which the string is to be inserted.</td> </tr> </table> | <i>str</i> | Is a pointer to the string to be inserted. | <i>n</i> | Is the number of characters not to exceed when inserting <i>str</i> . If <i>n</i> is less than 1, the entire string is inserted. | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string. | <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string. | <i>win</i> | Is a pointer to the window in which the string is to be inserted. |
| <i>str</i> | Is a pointer to the string to be inserted. | | | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when inserting <i>str</i> . If <i>n</i> is less than 1, the entire string is inserted. | | | | | | | | | | |
| <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string. | | | | | | | | | | |
| <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the string is to be inserted. | | | | | | | | | | |
| DESCRIPTION | <p>The <code>insstr()</code> function inserts <i>str</i> at the current cursor position of the <code>stdscr</code> window. The <code>winsstr()</code> function performs the identical action, but in window <i>win</i>. The <code>mvinsstr()</code> and <code>mvwinsstr()</code> functions insert the character string at the starting position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <i>win</i>).</p> <p>The <code>insnstr()</code>, <code>winsnstr()</code>, <code>mvinsnstr()</code>, and <code>mvwinsnstr()</code> functions insert <i>n</i> characters to the window or as many as will fit on the line. If <i>n</i> is less than 1, the entire string is inserted or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <i>x</i> and <i>y</i> parameters.</p> <p>All characters to the right of inserted characters are moved to the right. Characters that don't fit on the current line are discarded. The cursor is left at the point of insertion.</p> | | | | | | | | | | |

mvwinsstr(3XCURSES)

If a character in *str* is a newline, carriage return, backspace, or tab, the cursor is moved appropriately. The cursor is moved to the next tab stop for each tab character (by default, tabs are eight characters apart). If the character is a control character other than those previously mentioned, the character is inserted using $\^x$ notation, where *x* is a printable character. `clrtoeol(3XCURSES)` is automatically done before a newline.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addchstr(3XCURSES)`, `addstr(3XCURSES)`, `clrtoeol(3XCURSES)`, `ins_nwstr(3XCURSES)`, `insch(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_instr, instr, innstr, winstr, winnstr, mvinstr, mvinnstr, mvwinstr, mvwinnstr – get a string of characters from a curses window | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int instr(char *str); int innstr(char *str, int n); int winstr(WINDOW *win, char *str); int winnstr(WINDOW *win, char *str, int n); int mvinstr(int y, int x, char *str); int mvinnstr(int y, int x, char *str, int n); int mvwinstr(WINDOW *win, int y, int x, char *str); int mvwinnstr(WINDOW *win, int y, int x, char *str, int n);</pre> | | | | |
| DESCRIPTION | These routines return a string of characters in <i>str</i> , starting at the current cursor position in the named window and ending at the right margin of the window. Attributes are stripped from the characters. The four functions with <i>n</i> as the last argument return the string at most <i>n</i> characters long. | | | | |
| RETURN VALUES | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), attributes(5) | | | | |
| NOTES | <p>The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.</p> <p>Note that all routines except winnstr() may be macros.</p> | | | | |

mvwinstr(3XCURSES)

| | | | | | | | | | | | |
|--------------------|--|------------|---|----------|--|----------|--|----------|---|------------|--|
| NAME | innstr, instr, mvinnstr, mvinstr, mvwinnstr, mvwinstr, winnstr, winstr – retrieve a multibyte character string (without rendition) | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int innstr(char *str, int n); int instr(char *str); int mvinnstr(int y, int x, char *str, int n); int mvinstr(int y, int x, char *str); int mvwinnstr(WINDOW *win, int y, int x, char *str, int n); int mvwinstr(WINDOW *win, int y, int x, char *str); int winstr(WINDOW *win, char *str); int winnstr(WINDOW *win, char *str, int n);</pre> | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>str</i></td><td>Is a pointer to an object that can hold the retrieved multibyte character string.</td></tr><tr><td><i>n</i></td><td>Is the number of characters not to exceed when retrieving <i>str</i>.</td></tr><tr><td><i>y</i></td><td>Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved.</td></tr><tr><td><i>x</i></td><td>Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved.</td></tr><tr><td><i>win</i></td><td>Is a pointer to the window in which the string is to be retrieved.</td></tr></table> | <i>str</i> | Is a pointer to an object that can hold the retrieved multibyte character string. | <i>n</i> | Is the number of characters not to exceed when retrieving <i>str</i> . | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | <i>win</i> | Is a pointer to the window in which the string is to be retrieved. |
| <i>str</i> | Is a pointer to an object that can hold the retrieved multibyte character string. | | | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when retrieving <i>str</i> . | | | | | | | | | | |
| <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | | | | | | | | | | |
| <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the string is to be retrieved. | | | | | | | | | | |
| DESCRIPTION | <p>The <code>instr()</code> and <code>winstr()</code> functions retrieve a multibyte character string (without attributes) starting at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively, and ending at the right margin. The <code>mvinstr()</code> and <code>mvwinstr()</code> functions retrieve a multibyte character string located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>innstr()</code>, <code>winnstr()</code>, <code>mvinnstr()</code>, and <code>mvwinnstr()</code> functions retrieve at most <i>n</i> characters from the window <code>stdscr</code> and <i>win</i>, respectively. The former two functions retrieve the string starting at the current cursor position; the latter two commands return the string, starting at the position specified by the <i>x</i> and <i>y</i> parameters.</p> | | | | | | | | | | |

mvwinstr(3XCURSES)

All these functions store the retrieved string in the object pointed to by *str*. They only store complete multibyte characters. If the area pointed to by *str* is not large enough to hold at least one character, these functions fail.

Only the character portion of the character/rendition pair is returned. To return the complete character/rendition pair, use `winchstr()`.

ERRORS OK Successful completion.

ERR An error occurred.

USAGE All functions except `winnstr()` may be macros.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `inch(3XCURSES)`, `inchstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

mvwinswch(3CURSES)

| NAME | <code> curs_inswch, inswch, winswch, mvinswch, mvwinswch </code> – insert a <code>wchar_t</code> character before the character under the cursor in a curses window | | | | |
|---------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int inswch(chtype <i>wch</i>); int winswch(WINDOW *<i>win</i>, chtype <i>wch</i>); int mvinswch(int <i>y</i>, int <i>x</i>, chtype <i>wch</i>); int mvwinswch(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>, chtype <i>wch</i>);</pre> | | | | |
| DESCRIPTION | These routines insert the character <i>wch</i> , holding a <code>wchar_t</code> character, before the character under the cursor. All characters to the right of the cursor are moved one space to the right, with the possibility of the rightmost character on the line being lost. The cursor position does not change (after moving to <i>y</i> , <i>x</i> , if specified). (This does not imply use of the hardware insert character feature.) | | | | |
| RETURN VALUE | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header file <code><curses.h></code> automatically includes the header files <code><stdio.h></code> , <code><unctrl.h></code> and <code><wider.h></code> . Note that <code>inswch()</code> , <code>mvinswch()</code> , and <code>mvwinswch()</code> may be macros. None of these routines can use the color attribute in <code>chtype</code> . | | | | |

| | | | | | | | | | |
|----------------------|--|------------|--|----------|---|----------|--|------------|--|
| NAME | ins_wch, wins_wch, mvins_wch, mvwins_wch – insert a complex character | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int ins_wch(const cchar_t *wch); int mvins_wch(int y, int x, const cchar_t *wch); int mvwins_wch(WINDOW *win, int y, int x, const cchar_t *wch); int wins_wch(WINDOW *win, const cchar_t *wch);</pre> | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>wch</i></td> <td>Is the complex character to be inserted.</td> </tr> <tr> <td><i>y</i></td> <td>Is the y (row) coordinate of the position of the character.</td> </tr> <tr> <td><i>x</i></td> <td>Is the x (column) coordinate of the position of the character.</td> </tr> <tr> <td><i>win</i></td> <td>Is a pointer to the window in which the character is to be inserted.</td> </tr> </table> | <i>wch</i> | Is the complex character to be inserted. | <i>y</i> | Is the y (row) coordinate of the position of the character. | <i>x</i> | Is the x (column) coordinate of the position of the character. | <i>win</i> | Is a pointer to the window in which the character is to be inserted. |
| <i>wch</i> | Is the complex character to be inserted. | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of the position of the character. | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of the position of the character. | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the character is to be inserted. | | | | | | | | |
| DESCRIPTION | <p>The <code>ins_wch()</code> function inserts the complex character <i>wch</i> at the current cursor position of the <code>stdscr</code> window. The <code>wins_wch()</code> function performs the identical action but in window <i>win</i>. The <code>mvins_wch()</code> and <code>mvwins_wch()</code> functions insert the character at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>). The cursor position does not change.</p> <p>All characters to the right of the inserted character are moved right one character. The last character on the line is deleted.</p> <p>Insertions and deletions occur at the character level. The cursor is adjusted to the first column of the character prior to the the operation.</p> | | | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | | | |
| ERRORS | None. | | | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | | | |

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

mvwins_wch(3XCURSES)

SEE ALSO | add_wch(3XCURSES), ins_nwstr(3XCURSES), libcurses(3XCURSES),
attributes(5), standards(5)

| NAME | curs_inswstr, inswstr, insnwstr, winswstr, winsnwstr, mvinswstr, mvinsnwstr, mvwinswstr, mvwinsnwstr – insert <code>wchar_t</code> string before character under the cursor in a curses window | | | | |
|---------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int inswstr(wchar_t *wstr); int insnwstr(wchar_t *wstr, int n); int winswstr(WINDOW *win, wchar_t *wstr); int winsnwstr(WINDOW *win, wchar_t *wstr, int n); int mvinswstr(int y, int x, wchar_t *wstr); int mvinsnwstr(int y, int x, wchar_t *wstr, int n); int mvwinswstr(WINDOW *win, int y, int x, wchar_t *wstr); int mvwinsnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);</pre> | | | | |
| DESCRIPTION | <p>These routines insert a <code>wchar_t</code> character string (as many <code>wchar_t</code> characters as will fit on the line) before the character under the cursor. All characters to the right of the cursor are moved to the right, with the possibility of the rightmost characters on the line being lost. The cursor position does not change (after moving to y, x, if specified). (This does not imply use of the hardware insert character feature.) The four routines with n as the last argument insert at most n <code>wchar_t</code> characters. If $n \leq 0$, then the entire string is inserted.</p> <p>If a character in <i>wstr</i> is a tab, newline, carriage return, or backspace, the cursor is moved appropriately within the window. A newline also does a <code>clrtoeol(3CURSES)</code> before moving. Tabs are considered to be at every eighth column. If a character in <i>wstr</i> is another control character, it is drawn in the $\wedge X$ notation. Calling <code>winwch(3CURSES)</code> after adding a control character (and moving to it, if necessary) does not return the control character, but instead returns the representation of the control character.</p> | | | | |
| RETURN VALUE | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>clrtoeol(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>winwch(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header file <code><curses.h></code> automatically includes the header files <code><stdio.h></code> , <code><unctrl.h></code> and <code><wchar.h></code> . | | | | |

`mvwinswstr(3CURSES)`

Note that all but `winswstr()` may be macros.

| | | | | | | | | | | | |
|--------------------|---|-------------|--|----------|---|----------|--|----------|---|------------|---|
| NAME | ins_nwstr, ins_wstr, mvins_nwstr, mvins_wstr, mvwins_nwstr, mvwins_wstr, wins_nwstr, wins_wstr – insert a wide character string | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int ins_nwstr(const wchar_t *wstr, int n); int ins_wstr(const wchar_t *wstr); int mvins_nwstr(int y, int x, const wchar_t *wstr, int n); int mvins_wstr(int y, int x, const wchar_t *wstr); int mvwins_nwstr(WINDOW *win, int y, int x, const wchar_t *wstr, int n); int mvwins_wstr(WINDOW *win, int y, int x, const wchar_t *wstr); int wins_nwstr(WINDOW *win, const wchar_t *wstr, int n); int wins_wstr(WINDOW *win, const wchar_t *wstr);</pre> | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>wstr</i></td> <td>Is a pointer to the string to be inserted.</td> </tr> <tr> <td><i>n</i></td> <td>Is the number of characters not to exceed when inserting <i>wstr</i>. If <i>n</i> is less than 1, the entire string is inserted.</td> </tr> <tr> <td><i>y</i></td> <td>Is the <i>y</i> (row) coordinate of the starting position of the string.</td> </tr> <tr> <td><i>x</i></td> <td>Is the <i>x</i> (column) coordinate of the starting position of the string.</td> </tr> <tr> <td><i>win</i></td> <td>Is a pointer to the window in which the string is to be inserted.</td> </tr> </table> | <i>wstr</i> | Is a pointer to the string to be inserted. | <i>n</i> | Is the number of characters not to exceed when inserting <i>wstr</i> . If <i>n</i> is less than 1, the entire string is inserted. | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string. | <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string. | <i>win</i> | Is a pointer to the window in which the string is to be inserted. |
| <i>wstr</i> | Is a pointer to the string to be inserted. | | | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when inserting <i>wstr</i> . If <i>n</i> is less than 1, the entire string is inserted. | | | | | | | | | | |
| <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string. | | | | | | | | | | |
| <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the string is to be inserted. | | | | | | | | | | |
| DESCRIPTION | <p>The <code>ins_wstr()</code> function inserts <i>wstr</i> at the current cursor position of the <code>stdscr</code> window. The <code>wins_wstr()</code> function performs the identical action, but in window <i>win</i>. The <code>mvins_wstr()</code> and <code>mvwins_wstr()</code> functions insert <i>wstr</i> string at the starting position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>ins_nwstr()</code>, <code>wins_nwstr()</code>, <code>mvins_nwstr()</code>, and <code>mvwins_nwstr()</code> functions insert <i>n</i> characters to the window or as many as will fit on the line. If <i>n</i> is less than 1, the entire string is inserted or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <i>x</i> and <i>y</i> parameters.</p> | | | | | | | | | | |

mvwins_wstr(3XCURSES)

All characters to the right of inserted characters are moved to the right. Characters that don't fit on the current line are discarded. The cursor is left at the point of insertion.

If a character in *wstr* is a newline, carriage return, backspace, or tab, the cursor is moved appropriately. The cursor is moved to the next tab stop for each tab character (by default, tabs are eight characters apart). If the character is a control character other than those previously mentioned, the character is inserted using \hat{x} notation, where x is a printable character. `clrtoeol(3XCURSES)` is automatically done before a newline.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `add_wchnstr(3XCURSES)`, `addnwstr(3XCURSES)`, `clrtoeol(3XCURSES)`, `ins_wch(3XCURSES)`, `insnstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

NAME curs_inwch, inwch, winwch, mvwinwch, mvwinwch – get a wchar_t character and its attributes from a curses window

SYNOPSIS

```
cc [ flag ... ] file ... -lcurses [ library .. ]
#include <curses.h>

ctype inwch(void);
ctype winwch(WINDOW *win);
ctype mvwinwch(int y, int x);
ctype mvwinwch(WINDOW *win, int y, int x);
```

DESCRIPTION These routines return the wchar_t character, of type ctype, at the current position in the named window. If any attributes are set for that position, their values are OR-ed into the value returned. Constants defined in <curses.h> can be used with the logical AND (&) operator to extract the character or attributes alone.

Attributes The following bit-masks may be AND-ed with characters returned by winwch().

```
A_WCHARTEXT      Bit-mask to extract character
A_WATTRIBUTES    Bit-mask to extract attributes
```

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), attributes(5)

NOTES The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <widec.h>.

Note that all of these routines may be macros.

None of these routines can use the color attribute in ctype.

mvwin_wch(3XCURSES)

NAME | in_wch, mvin_wch, mvwin_wch, win_wch – retrieve a complex character (with rendition)

SYNOPSIS | `cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \`
`-R /usr/xpg4/lib -lcurses [library...]`

`c89 [flag...] file... -lcurses [library...]`

`#include <curses.h>`

`int in_wch(cchar_t *wcv);`

`int mvin_wch(int y, int x, cchar_t *wcv);`

`int mvwin_wch(WINDOW *win, int y, cchar_t *wcv);`

`int win_wch(WINDOW *win, cchar_t *wcv);`

DESCRIPTION | The in_wch() and win_wch() functions retrieve the complex character and its rendition located at the current cursor position of the stdscr window and window win, respectively. The mvin_wch() and mvwin_wch() functions retrieve the complex character and its rendition located at the position indicated by the x (column) and y (row) parameters (the former in the stdscr window; the latter in window win).

All these functions store the retrieved character and its rendition in the object pointed to by wcv.

PARAMETERS | *wcv* | Is a pointer to an object that can store a complex character and its rendition.

| *y* | Is the y (row) coordinate of the position of the character to be returned.

| *x* | Is the x (column) coordinate of the position of the character to be returned.

| *win* | Is a pointer to the window that contains the character to be returned.

RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR.

ERRORS | None.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | add_wch(3XCURSES), inch(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

| NAME | curs_inwchstr, inwchstr, inwchnstr, winwchstr, winwchnstr, mvinwchstr, mvinwchnstr, mvwinwchstr, mvwinwchnstr – get a string of wchar_t characters (and attributes) from a curses window | | | | |
|---------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int inwchstr(chtype *wchstr); int inwchnstr(chtype *wchstr, int n); int winwchstr(WINDOW *win, chtype *wchstr); int winwchnstr(WINDOW *win, chtype *wchstr, int n); int mvinwchstr(int y, int x, chtype *wchstr); int mvinwchnstr(int y, int x, chtype *wchstr, int n); int mvwinwchstr(WINDOW *win, int y, int x, chtype *wchstr); int mvwinwchnstr(WINDOW *win, int y, int x, chtype *wchstr, int n);</pre> | | | | |
| DESCRIPTION | These routines return a string of type chtype, holding wchar_t characters, starting at the current cursor position in the named window and ending at the right margin of the window. The four functions with <i>n</i> as the last argument, return the string at most <i>n</i> wchar_t characters long. Constants defined in <curses.h> can be used with the logical AND (&) operator to extract the wchar_t character or the attribute alone from any position in the <i>wchstr</i> (see curs_inwch(3CURSES)). | | | | |
| RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See attributes(5) for a description of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), curs_inwch(3CURSES), attributes(5) | | | | |
| NOTES | <p>The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <widec.h>.</p> <p>Note that all routines except winwchnstr() may be macros.</p> <p>None of these routines can use the color attribute in chtype.</p> | | | | |

mvwin_wchnstr(3XCURSES)

| | | | | | | | | | | | |
|--------------------|---|---------------|---|----------|---|----------|--|----------|---|------------|--|
| NAME | in_wchnstr, in_wchstr, mvin_wchnstr, mvin_wchstr, mvwin_wchnstr, mvwin_wchstr, win_wchnstr, win_wchstr – retrieve complex character string (with rendition) | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int in_wchnstr(cchar_t *wchstr, int n); int in_wchstr(cchar_t *wchstr); int mvin_wchnstr(int y, int x, cchar_t *wchstr, int n); int mvin_wchstr(int y, int x, cchar_t *wchstr); int mvwin_wchnstr(WINDOW *win, int y, int x, cchar_t *wchstr, int n); int mvwin_wchstr(WINDOW *win, int y, int x, cchar_t *wchstr); int win_wchnstr(WINDOW *win, cchar_t *wchstr, int n); int win_wchstr(WINDOW *win, cchar_t *wchstr);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The <code>in_wchstr()</code> and <code>win_wchstr()</code> functions retrieve a complex character string (with rendition) starting at the current cursor position of the <code>stdscr</code> window and window <code>win</code>, respectively, and ending at the right margin. The <code>mvin_wchstr()</code> and <code>mvwin_wchstr()</code> functions retrieve a complex character string located at the position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <code>win</code>).</p> <p>The <code>in_wchnstr()</code>, <code>win_wchnstr()</code>, <code>mvin_wchnstr()</code>, and <code>mvwin_wchnstr()</code> functions retrieve at most <code>n</code> characters from the window <code>stdscr</code> and <code>win</code>, respectively. The former two functions retrieve the string, starting at the current cursor position; the latter two commands retrieve the string, starting at the position specified by the <code>x</code> and <code>y</code> parameters.</p> <p>The retrieved character string (with renditions) is stored in the object pointed to by <code>wcval</code>.</p> | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>wchstr</i></td> <td>Is a pointer to an object where the retrieved complex character string can be stored.</td> </tr> <tr> <td><i>n</i></td> <td>Is the number of characters not to exceed when retrieving <i>wchstr</i>.</td> </tr> <tr> <td><i>y</i></td> <td>Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved.</td> </tr> <tr> <td><i>x</i></td> <td>Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved.</td> </tr> <tr> <td><i>win</i></td> <td>Is a pointer to the window in which the string is to be retrieved.</td> </tr> </table> | <i>wchstr</i> | Is a pointer to an object where the retrieved complex character string can be stored. | <i>n</i> | Is the number of characters not to exceed when retrieving <i>wchstr</i> . | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | <i>win</i> | Is a pointer to the window in which the string is to be retrieved. |
| <i>wchstr</i> | Is a pointer to an object where the retrieved complex character string can be stored. | | | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when retrieving <i>wchstr</i> . | | | | | | | | | | |
| <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | | | | | | | | | | |
| <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the string is to be retrieved. | | | | | | | | | | |

mvwin_wchnstr(3XCURSES)

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `in_wch(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

mvwinchstr(3CURSES)

| NAME | <code> curs_inchstr, inchstr, inchnstr, winchstr, winchnstr, mvinchstr, mvinchnstr, mvwinchstr, mvwinchnstr</code> – get a string of <code>wchar_t</code> characters (and attributes) from a curses window | | | | |
|---------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int inchstr(cctype *wchstr); int inchnstr(cctype *wchstr, int n); int winchstr(WINDOW *win, cctype *wchstr); int winchnstr(WINDOW *win, cctype *wchstr, int n); int mvinchstr(int y, int x, cctype *wchstr); int mvinchnstr(int y, int x, cctype *wchstr, int n); int mvwinchstr(WINDOW *win, int y, int x, cctype *wchstr); int mvwinchnstr(WINDOW *win, int y, int x, cctype *wchstr, int n);</pre> | | | | |
| DESCRIPTION | These routines return a string of type <code>cctype</code> , holding <code>wchar_t</code> characters, starting at the current cursor position in the named window and ending at the right margin of the window. The four functions with <code>n</code> as the last argument, return the string at most <code>n</code> <code>wchar_t</code> characters long. Constants defined in <code><curses.h></code> can be used with the logical AND (<code>&</code>) operator to extract the <code>wchar_t</code> character or the attribute alone from any position in the <code>wchstr</code> (see <code>curs_inch(3CURSES)</code>). | | | | |
| RETURN VALUE | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for a description of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>curs_inch(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header file <code><curses.h></code> automatically includes the header files <code><stdio.h></code> , <code><unctrl.h></code> and <code><wdec.h></code> . Note that all routines except <code>winchnstr()</code> may be macros. None of these routines can use the <code>color</code> attribute in <code>cctype</code> . | | | | |

| | | | | | | | | | | | |
|--------------------|---|---------------|---|----------|---|----------|--|----------|---|------------|--|
| NAME | in_wchnstr, in_wchstr, mvin_wchnstr, mvin_wchstr, mvwin_wchnstr, mvwin_wchstr, win_wchnstr, win_wchstr – retrieve complex character string (with rendition) | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int in_wchnstr(cchar_t *wchstr, int n); int in_wchstr(cchar_t *wchstr); int mvin_wchnstr(int y, int x, cchar_t *wchstr, int n); int mvin_wchstr(int y, int x, cchar_t *wchstr); int mvwin_wchnstr(WINDOW *win, int y, int x, cchar_t *wchstr, int n); int mvwin_wchstr(WINDOW *win, int y, int x, cchar_t *wchstr); int win_wchnstr(WINDOW *win, cchar_t *wchstr, int n); int win_wchstr(WINDOW *win, cchar_t *wchstr);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The <code>in_wchstr()</code> and <code>win_wchstr()</code> functions retrieve a complex character string (with rendition) starting at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively, and ending at the right margin. The <code>mvin_wchstr()</code> and <code>mvwin_wchstr()</code> functions retrieve a complex character string located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>in_wchnstr()</code>, <code>win_wchnstr()</code>, <code>mvin_wchnstr()</code>, and <code>mvwin_wchnstr()</code> functions retrieve at most <i>n</i> characters from the window <code>stdscr</code> and <i>win</i>, respectively. The former two functions retrieve the string, starting at the current cursor position; the latter two commands retrieve the string, starting at the position specified by the <i>x</i> and <i>y</i> parameters.</p> <p>The retrieved character string (with renditions) is stored in the object pointed to by <i>wcval</i>.</p> | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>wchstr</i></td> <td>Is a pointer to an object where the retrieved complex character string can be stored.</td> </tr> <tr> <td><i>n</i></td> <td>Is the number of characters not to exceed when retrieving <i>wchstr</i>.</td> </tr> <tr> <td><i>y</i></td> <td>Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved.</td> </tr> <tr> <td><i>x</i></td> <td>Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved.</td> </tr> <tr> <td><i>win</i></td> <td>Is a pointer to the window in which the string is to be retrieved.</td> </tr> </table> | <i>wchstr</i> | Is a pointer to an object where the retrieved complex character string can be stored. | <i>n</i> | Is the number of characters not to exceed when retrieving <i>wchstr</i> . | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | <i>win</i> | Is a pointer to the window in which the string is to be retrieved. |
| <i>wchstr</i> | Is a pointer to an object where the retrieved complex character string can be stored. | | | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when retrieving <i>wchstr</i> . | | | | | | | | | | |
| <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | | | | | | | | | | |
| <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the string is to be retrieved. | | | | | | | | | | |

mvwin_wchstr(3XCURSES)

RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR.

ERRORS | None.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | `in_wch(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

NAME curs_inwstr, inwstr, innwstr, winwstr, winnwstr, mvinwstr, mvinnwstr, mvwinwstr, mvwinnwstr – get a string of wchar_t characters from a curses window

SYNOPSIS

```
cc [ flag ... ] file ... -lcurses[library .. ]
#include <curses.h>

int inwstr(wchar_t *wstr);
int innwstr(wchar_t *wstr, int n);
int winwstr(WINDOW *win, wchar_t *wstr);
int winnwstr(WINDOW *win, wchar_t *wstr, int n);
int mvinwstr(int y, int x, wchar_t *wstr);
int mvinnwstr(int y, int x, wchar_t *wstr, int n);
int mvwinwstr(WINDOW *win, int y, int x, wchar_t *wstr);
int mvwinnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);
```

DESCRIPTION These routines return the string of wchar_t characters in *wstr* starting at the current cursor position in the named window and ending at the right margin of the window. Attributes are stripped from the characters. The four functions with *n* as the last argument return the string at most *n* wchar_t characters long.

RETURN VALUES All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), attributes(5)

NOTES The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <wdec.h>.

Note that all routines except winnwstr() may be macros.

mvwinwstr(3XCURSES)

| | |
|--------------------|---|
| NAME | <code>innwstr</code> , <code>inwstr</code> , <code>mvinnwstr</code> , <code>mvinwstr</code> , <code>mvwinnwstr</code> , <code>mvwinwstr</code> , <code>winnwstr</code> , <code>winwstr</code> – retrieve a wide character string (without rendition) |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int innwstr(wchar_t *wstr, int n); int inwstr(wchar_t *wstr); int mvinnwstr(int y, int x, wchar_t *wstr, int n); int mvinwstr(int y, int x, wchar_t *wstr); int mvwinnwstr(WINDOW*win, int y, int x, wchar_t *wstr, int n); int mvwinwstr(WINDOW*win, int y, int x, wchar_t *wstr); int winwstr(WINDOW*win, wchar_t *wstr); int winnwstr(WINDOW*win, wchar_t *wstr, int n);</pre> |
| PARAMETERS | <p><i>wstr</i> Is a pointer to an object that can hold the retrieved multibyte character string.</p> <p><i>n</i> Is the number of characters not to exceed when retrieving <i>wstr</i>.</p> <p><i>y</i> Is the y (row) coordinate of the starting position of the string to be retrieved.</p> <p><i>x</i> Is the x (column) coordinate of the starting position of the string to be retrieved.</p> <p><i>win</i> Is a pointer to the window in which the string is to be retrieved.</p> |
| DESCRIPTION | <p>The <code>inwstr()</code> and <code>winwstr()</code> functions retrieve a wide character string (without attributes) starting at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively, and ending at the right margin. The <code>mvinwstr()</code> and <code>mvwinwstr()</code> functions retrieve a wide character string located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>innwstr()</code>, <code>winnwstr()</code>, <code>mvinnwstr()</code>, and <code>mvwinnwstr()</code> functions retrieve at most <i>n</i> characters from the window <code>stdscr</code> and <i>win</i>, respectively. The former two functions retrieve the string starting at the current cursor position; the latter two commands return the string, starting at the position specified by the <i>x</i> and <i>y</i> parameters.</p> <p>All these functions store the retrieved string in the object pointed to by <i>wstr</i>. They only store complete wide characters. If the area pointed to by <i>wstr</i> is not large enough to hold at least one character, these functions fail.</p> |

`mvwinwstr(3XCURSES)`

Only the character portion of the character/rendition pair is returned. To return the complete character/rendition pair, use `win_wchstr(3XCURSES)`.

RETURN VALUES

On success, the `inwstr()`, `mvinwstr()`, `mvwinwstr()`, and `winwstr()` functions return OK. Otherwise, they return ERR.

On success, the `innwstr()`, `mvinnwstr()`, `mvwinnwstr()`, and `winnwstr()` functions return the number of characters read into the string. Otherwise, they return ERR.

ERRORS

None.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO

`in_wch(3XCURSES)`, `in_wchnstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

mvwprintw(3CURSES)

| NAME | <code> curs_</code> printw, <code>printw</code> , <code>wprintw</code> , <code>mvprintw</code> , <code>mvwprintw</code> , <code>vwprintw</code> – print formatted output in curses windows | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int printw(char *<i>fmt</i>, /* <i>arg</i> */ ...); int wprintw(WINDOW *<i>win</i>, char *<i>fmt</i>, /* <i>arg</i> */ ...); int mvprintw(int <i>y</i>, int <i>x</i>, char *<i>fmt</i>, /* <i>arg</i> */ ...); int mvwprintw(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>, char *<i>fmt</i>, /* <i>arg</i> */...); #include <varargs.h> int vwprintw(WINDOW *<i>win</i>, char *<i>fmt</i>, /* <i>varglist</i> */ ...);</pre> | | | | |
| DESCRIPTION | <p>The <code>printw()</code>, <code>wprintw()</code>, <code>mvprintw()</code>, and <code>mvwprintw()</code> routines are analogous to <code>printf()</code> (see <code>printf(3C)</code>). In effect, the string that would be output by <code>printf()</code> is output instead as though <code>waddstr()</code> were used on the given window.</p> <p>The <code>vwprintw()</code> routine is analogous to <code>vprintf()</code> (see <code>vprintf(3C)</code>) and performs a <code>wprintw()</code> using a variable argument list. The third argument is a <code>va_list</code>, a pointer to a list of arguments, as defined in <code><varargs.h></code>.</p> | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>printf(3C)</code> , <code>vprintf(3C)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . | | | | |

| NAME | mvprintw, mvwprintw, printw, wprintw – print formatted output window | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int mvprintw(int y, int x, char *fmt, ...); int mvwprintw(WINDOW *win, int y, int x, char *fmt, ...); int printw(char *fmt, ...); int wprintw(WINDOW *win, char *fmt, ...);</pre> | | | | | | |
| PARAMETERS | <p><i>y</i> Is the y (row) coordinate position of the string's placement in the window.</p> <p><i>x</i> Is the x (column) coordinate position of the string's placement in the window.</p> <p><i>fmt</i> Is a printf() format string.</p> <p><i>win</i> Is a pointer to the window in which the string is to be written.</p> | | | | | | |
| DESCRIPTION | The mvprintw(), mvwprintw(), printw(), and wprintw() functions are analogous to printf(3C). The effect of these functions is as though sprintf() were used to format the string, and then waddstr(3XCURSES) were used to add that multi-byte string to the current or specified window at the current or specified cursor position. | | | | | | |
| RETURN VALUES | Upon successful completion, these functions return OK. Otherwise, they return ERR. | | | | | | |
| ERRORS | No errors are defined. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | addnstr(3XCURSES), libcurses(3XCURSES), printf(3C), attributes(5), standards(5) | | | | | | |

mvwscanw(3CURSES)

| NAME | curs_scanw, scanw, wscanw, mvscanw, mvwscanw, vwscanw – convert formatted input from a curses widow | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int scanw(char *fmt, /* arg */ ...); int wscanw(WINDOW *win, char *fmt, /* arg */ ...); int mvscanw(int y, int x, char *fmt, /* arg */ ...); int mvwscanw(WINDOW *win, int y, int x, char *fmt, /* arg */...); int vwscanw(WINDOW *win, char *fmt, va_list varglist);</pre> | | | | |
| DESCRIPTION | <p>The scanw(), wscanw(), and mvscanw() routines correspond to scanf() (see scanf(3C)). The effect of these routines is as though wgetstr() were called on the window, and the resulting line used as input for the scan. Fields which do not map to a variable in the fmt field are lost.</p> <p>The vwscanw() routine is similar to vwprintw() in that it performs a wscanw() using a variable argument list. The third argument is a va_list, a pointer to a list of arguments, as defined in <varargs.h>.</p> | | | | |
| RETURN VALUES | <p>vwscanw() returns ERR on failure and an integer equal to the number of fields scanned on success.</p> <p>Applications may interrogate the return value from the scanw, wscanw(), mvscanw(), and mvwscanw() routines to determine the number of fields which were mapped in the call.</p> | | | | |
| ATTRIBUTES | <p>See attributes(5) for descriptions of the following attributes:</p> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">MT-Level</td> <td style="text-align: center;">Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curs_getstr(3CURSES), curs_printw(3CURSES), curses(3CURSES), scanf(3C), attributes(5) | | | | |
| NOTES | The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>. | | | | |

NAME mvscanw, mvwscanw, scanw, wscanw – convert formatted input from a window

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

int mvscanw(int y, int x, char *fmt, ...);
int mvwscanw(WINDOW *win, int y, int x, char *fmt, ...);
int scanw(char *fmt, ...);
int wscanw(WINDOW *win, char *fmt, ...);
```

PARAMETERS

y Is the y (row) coordinate of the position of the character to be read.

x Is the x (column) coordinate of the position of the character to be read.

fmt Is a scanf () format string.

win Is a pointer to the window in which the character is to be read.

DESCRIPTION These functions are similar to scanf(3C). Their effect is as though mvwgetstr(3XCURSES) were called to get a multi-byte character string from the current or specified window at the current or specified cursor position, and then sscanf () were used to interpret and convert that string.

RETURN VALUES Upon successful completion, these functions return OK. Otherwise, they return ERR.

ERRORS No errors are defined.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO getnstr(3XCURSES), libcurses(3XCURSES), printw(3XCURSES), scanf(3C), wcstombs(3C), attributes(5), standards(5)

mvwvline(3XCURSES)

| | |
|----------------------|--|
| NAME | <code>hline</code> , <code>mvhline</code> , <code>mvvline</code> , <code>mvwhline</code> , <code>mvwvline</code> , <code>vline</code> , <code>whline</code> , <code>wvline</code> – use single-byte characters (and renditions) to draw lines |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int hline(chtype ch, int n); int mvhline(int y, int x, chtype ch, int n); int mvvline(int y, int x, chtype ch, int n); int mvwhline(WINDOW *win, int y, int x, chtype ch, int n); int mvwvline(WINDOW *win, int y, int x, chtype ch, int n); int vline(chtype ch, int n); int whline(WINDOW *win, chtype ch, int n); int wvline(WINDOW *win, chtype ch, int n);</pre> |
| DESCRIPTION | <p>The <code>hline()</code>, <code>vline()</code>, <code>whline()</code>, <code>wvline()</code> functions draw a horizontal or vertical line, in either the window <code>stdscr</code> or <code>win</code> starting at the current cursor position. The line is drawn using the character <code>ch</code> and is a maximum of <code>n</code> positions long, or as many as will fit into the window. If <code>ch</code> is 0 (zero), the default horizontal or vertical character is used.</p> <p>The <code>mvhline()</code>, <code>mvvline()</code>, <code>mvwhline()</code>, <code>mvwvline()</code> functions are similar to the previous group of functions but the line begins at cursor position specified by <code>x</code> and <code>y</code>.</p> <p>The functions with names ending with <code>hline()</code> draw horizontal lines proceeding towards the last column of the same line. The functions with names ending with <code>vline()</code> draw vertical lines proceeding towards the last column of the same line.</p> <p>These functions do not change the position of the cursor.</p> |
| PARAMETERS | <p><code>ch</code> Is the character used to draw the line.</p> <p><code>n</code> Is the maximum number of characters in the line.</p> <p><code>y</code> Is the <code>y</code> (row) coordinate for the start of the line.</p> <p><code>x</code> Is the <code>x</code> (column) coordinate for the start of the line.</p> <p><code>win</code> Is a pointer to a window.</p> |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. |
| ERRORS | None |

mvwvline(3XCURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `border(3XCURSES)`, `border_set(3XCURSES)`, `hline_set(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

mvwvline_set(3XCURSES)

| | | | | | | | | | | | |
|----------------------|---|------------------|---|----------------|--|----------------|--|----------------|---|------------------|---------------------------|
| NAME | <code>hline_set</code> , <code>mvhline_set</code> , <code>mvvline_set</code> , <code>mvwhline_set</code> , <code>mvwvline_set</code> , <code>vline_set</code> , <code>whline_set</code> , <code>wvline_set</code> – use complex characters (and renditions) to draw lines | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int hline_set(const cchar_t *ch, int n); int mvhline_set(int y, int x, const cchar_t *wch, int n); int mvvline_set(int y, int x, const cchar_t *wch, int n); int mvwhline_set(WINDOW *win, int y, int x, const cchar_t *wch, int n); int mvwvline_set(WINDOW *win, int y, int x, const cchar_t *wch, int n); int vline_set(const cchar_t *wch, int n); int whline_set(WINDOW *win, const cchar_t *wch, int n); int wvline_set(WINDOW *win, const cchar_t *wch, int n);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The <code>hline_set()</code>, <code>vline_set()</code>, <code>whline_set()</code>, <code>wvline_set()</code> functions draw a line, in either the window <code>stdscr</code> or <code>win</code> starting at the current cursor position. The line is drawn using the character <code>wch</code> and is a maximum of <code>n</code> positions long, or as many as will fit into the window. If <code>wch</code> is a null pointer, the default horizontal or vertical character is used.</p> <p>The <code>mvhline_set()</code>, <code>mvvline_set()</code>, <code>mvwhline_set()</code>, <code>mvwvline_set()</code> functions are similar to the previous group of functions but the line begins at cursor position specified by <code>x</code> and <code>y</code>.</p> <p>The functions with names ending with <code>hline_set()</code> draw horizontal lines proceeding towards the last column of the same line. The functions with names ending with <code>vline_set()</code> draw vertical lines proceeding towards the last column of the same line.</p> <p>These functions do not change the position of the cursor.</p> | | | | | | | | | | |
| PARAMETERS | <table><tr><td><code>wch</code></td><td>Is the complex character used to draw the line.</td></tr><tr><td><code>n</code></td><td>Is the maximum number of characters in the line.</td></tr><tr><td><code>y</code></td><td>Is the y (row) coordinate for the start of the line.</td></tr><tr><td><code>x</code></td><td>Is the x (column) coordinate for the start of the line.</td></tr><tr><td><code>win</code></td><td>Is a pointer to a window.</td></tr></table> | <code>wch</code> | Is the complex character used to draw the line. | <code>n</code> | Is the maximum number of characters in the line. | <code>y</code> | Is the y (row) coordinate for the start of the line. | <code>x</code> | Is the x (column) coordinate for the start of the line. | <code>win</code> | Is a pointer to a window. |
| <code>wch</code> | Is the complex character used to draw the line. | | | | | | | | | | |
| <code>n</code> | Is the maximum number of characters in the line. | | | | | | | | | | |
| <code>y</code> | Is the y (row) coordinate for the start of the line. | | | | | | | | | | |
| <code>x</code> | Is the x (column) coordinate for the start of the line. | | | | | | | | | | |
| <code>win</code> | Is a pointer to a window. | | | | | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | | | | | |

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `border(3XCURSES)`, `border_set(3XCURSES)`, `hline(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

napms(3CURSES)

| | |
|--------------------|--|
| NAME | <code> curs_kernel, def_prog_mode, def_shell_mode, reset_prog_mode, reset_shell_mode, resetty, savetty, getsyx, setsyx, ripoffline, curs_set, napms </code> – low-level curses routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int def_prog_mode(void); int def_shell_mode(void); int reset_prog_mode(void); int reset_shell_mode(void); int resetty(void); int savetty(void); int getsyx(int y, int x); int setsyx(int y, int x); int ripoffline(int line, int (*init)(WINDOW *, int)); int curs_set(int visibility); int napms(int ms);</pre> |
| DESCRIPTION | <p>The following routines give low-level access to various curses functionality. These routines typically are used inside library routines.</p> <p>The <code>def_prog_mode()</code> and <code>def_shell_mode()</code> routines save the current terminal modes as the “program” (in curses) or “shell” (not in curses) state for use by the <code>reset_prog_mode()</code> and <code>reset_shell_mode()</code> routines. This is done automatically by <code>initscr()</code>.</p> <p>The <code>reset_prog_mode()</code> and <code>reset_shell_mode()</code> routines restore the terminal to “program” (in curses) or “shell” (out of curses) state. These are done automatically by <code>endwin()</code> and, after an <code>endwin()</code>, by <code>doupdate()</code>, so they normally are not called.</p> <p>The <code>resetty()</code> and <code>savetty()</code> routines save and restore the state of the terminal modes. <code>savetty()</code> saves the current state in a buffer and <code>resetty()</code> restores the state to what it was at the last call to <code>savetty()</code>.</p> <p>With the <code>getsyx()</code> routine, the current coordinates of the virtual screen cursor are returned in <code>y</code> and <code>x</code>. If <code>leaveok()</code> is currently TRUE, then <code>-1,-1</code> is returned. If lines have been removed from the top of the screen, using <code>ripoffline()</code>, <code>y</code> and <code>x</code> include these lines; therefore, <code>y</code> and <code>x</code> should be used only as arguments for <code>setsyx()</code>.</p> <p>With the <code>setsyx()</code> routine, the virtual screen cursor is set to <code>y, x</code>. If <code>y</code> and <code>x</code> are both <code>-1</code>, then <code>leaveok()</code> is set. The two routines <code>getsyx()</code> and <code>setsyx()</code> are designed to be used by a library routine, which manipulates curses windows but does not</p> |

want to change the current position of the program's cursor. The library routine would call `getsyx()` at the beginning, do its manipulation of its own windows, do a `wnoutrefresh()` on its windows, call `setsyx()`, and then call `doupdate()`.

The `riporffline()` routine provides access to the same facility that `slk_init()` (see `curs_slk(3CURSES)`) uses to reduce the size of the screen. `riporffline()` must be called before `initscr()` or `newterm()` is called. If `line` is positive, a line is removed from the top of `stdscr()`; if `line` is negative, a line is removed from the bottom. When this is done inside `initscr()`, the routine `init()` (supplied by the user) is called with two arguments: a window pointer to the one-line window that has been allocated and an integer with the number of columns in the window. Inside this initialization routine, the integer variables `LINES` and `COLS` (defined in `<curses.h>`) are not guaranteed to be accurate and `wrefresh()` or `doupdate()` must not be called. It is allowable to call `wnoutrefresh()` during the initialization routine.

`riporffline()` can be called up to five times before calling `initscr()` or `newterm()`.

With the `curs_set()` routine, the cursor state is set to invisible, normal, or very visible for *visibility* equal to 0, 1, or 2 respectively. If the terminal supports the *visibility* requested, the previous *cursor* state is returned; otherwise, `ERR` is returned.

The `napms()` routine is used to sleep for *ms* milliseconds.

RETURN VALUES

Except for `curs_set()`, these routines always return `OK`. `curs_set()` returns the previous cursor state, or `ERR` if the requested *visibility* is not supported.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_initscr(3CURSES)`, `curs_outopts(3CURSES)`, `curs_refresh(3CURSES)`, `curs_scr_dump(3CURSES)`, `curs_slk(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `getsyx()` is a macro, so an ampersand (&) is not necessary before the variables *y* and *x*.

napms(3XCURSES)

NAME napms – sleep process for a specified length of time

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

int napms(int ms);
```

PARAMETERS *ms* Is the number of milliseconds to sleep.

DESCRIPTION The napms () function sleeps for at least *ms* milliseconds.

RETURN VALUES The napms () function always returns OK.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO delay_output(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

new_field(3CURSES)

NAME form_field_new, new_field, dup_field, link_field, free_field – create and destroy forms fields

SYNOPSIS

```
cc [ flag... ] file... -lform -lcurses [ library... ]
#include <form.h>

FIELD *new_field(int r, int c, int frow, int fcol, int nrow, int ncol);
FIELD *dup_field(FIELD *field, int frow, int fcol);
FIELD *link_field(FIELD *field, int frow, int fcol);
int free_field(FIELD *field);
```

DESCRIPTION new_field() creates a new field with *r* rows and *c* columns, starting at *frow*, *fcol*, in the subwindow of a form. *nrow* is the number of off-screen rows and *nbuf* is the number of additional working buffers. This routine returns a pointer to the new field.

dup_field() duplicates *field* at the specified location. All field attributes are duplicated, including the current contents of the field buffers.

link_field() also duplicates *field* at the specified location. However, unlike dup_field(), the new field shares the field buffers with the original field. After creation, the attributes of the new field can be changed without affecting the original field.

free_field() frees the storage allocated for *field*.

RETURN VALUES Routines that return pointers return NULL on error. free_field() returns one of the following:

| | |
|----------------|---|
| E_OK | The function returned successfully. |
| E_CONNECTED | The field is already connected to a form. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An argument is incorrect. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), forms(3CURSES), attributes(5)

NOTES The header <form.h> automatically includes the headers <eti.h> and <curses.h>.

new_fieldtype(3CURSES)

| | | | | | | | | | |
|----------------------|--|------|-------------------------------------|----------------|---------------|----------------|---------------------------|-------------|--|
| NAME | form_fieldtype, new_fieldtype, free_fieldtype, set_fieldtype_arg, set_fieldtype_choice, link_fieldtype – forms fieldtype routines | | | | | | | | |
| SYNOPSIS | <pre>cc [<i>flag...</i>] <i>file...</i> -lform -lcurses [<i>library...</i>] #include <form.h> FIELDTYPE *new_fieldtype(int (* <i>field_check</i>)(FIELD *, char *), int (*<i>char_check</i>)(int, char *)); int free_fieldtype(FIELDTYPE *<i>fieldtype</i>); int set_fieldtype_arg(FIELDTYPE *<i>fieldtype</i>, char *(* <i>mak_arg</i>)(va_list *), char *(* <i>copy_arg</i>)(char *), void (* <i>free_arg</i>)(char *)); int set_fieldtype_choice(FIELDTYPE *<i>fieldtype</i>, int (* <i>next_choice</i>)(FIELD *, char *), int (*<i>prev_choice</i>)(FIELD *, char *)); FIELDTYPE *link_fieldtype(FIELDTYPE *<i>type1</i>, FIELDTYPE *<i>type2</i>);</pre> | | | | | | | | |
| DESCRIPTION | <p>new_fieldtype() creates a new field type. The application programmer must write the function <i>field_check</i>, which validates the field value, and the function <i>char_check</i>, which validates each character. free_fieldtype() frees the space allocated for the field type.</p> <p>By associating function pointers with a field type, set_fieldtype_arg() connects to the field type additional arguments necessary for a set_field_type() call. Function <i>mak_arg</i> allocates a structure for the field specific parameters to set_field_type() and returns a pointer to the saved data. Function <i>copy_arg</i> duplicates the structure created by <i>make_arg</i>. Function <i>free_arg</i> frees any storage allocated by <i>make_arg</i> or <i>copy_arg</i>.</p> <p>The form_driver() requests REQ_NEXT_CHOICE and REQ_PREV_CHOICE let the user request the next or previous value of a field type comprising an ordered set of values. set_fieldtype_choice() allows the application programmer to implement these requests for the given field type. It associates with the given field type those application-defined functions that return pointers to the next or previous choice for the field.</p> <p>link_fieldtype() returns a pointer to the field type built from the two given types. The constituent types may be any application-defined or pre-defined types.</p> | | | | | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <table><tr><td>E_OK</td><td>The function returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr><tr><td>E_BAD_ARGUMENT</td><td>An argument is incorrect.</td></tr><tr><td>E_CONNECTED</td><td>Type is connected to one or more fields.</td></tr></table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An argument is incorrect. | E_CONNECTED | Type is connected to one or more fields. |
| E_OK | The function returned successfully. | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | |
| E_BAD_ARGUMENT | An argument is incorrect. | | | | | | | | |
| E_CONNECTED | Type is connected to one or more fields. | | | | | | | | |

`new_fieldtype(3CURSES)`

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `courses(3CURSES)`, `forms(3CURSES)`, `attributes(5)`

NOTES The header `<form.h>` automatically includes the headers `<eti.h>` and `<courses.h>`.

new_form(3CURSES)

NAME form_new, new_form, free_form – create and destroy forms

SYNOPSIS

```
cc [ flag... ] file... -lform -lcurses [ library... ]
#include <form.h>

FORM *new_form(FIELD **fields) ;
int free_form(FORM *form) ;
```

DESCRIPTION new_form() creates a new form connected to the designated fields and returns a pointer to the form.

free_form() disconnects the *form* from its associated field pointer array and deallocates the space for the form.

RETURN VALUES new_form() always returns NULL on error. free_form() returns one of the following:

| | |
|----------------|-------------------------------------|
| E_OK | The function returned successfully. |
| E_BAD_ARGUMENT | An argument is incorrect. |
| E_POSTED | The form is posted. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), forms(3CURSES), attributes(5)

NOTES The header <form.h> automatically includes the headers <eti.h> and <curses.h>.

| NAME | menu_item_new, new_item, free_item – create and destroy menu items | | | | | | | | |
|----------------------|---|----------------|------------------------------------|----------------|---------------|----------------|--|-------------|--|
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> ITEM *new_item(char *name, char *desc); int free_item(ITEM *item);</pre> | | | | | | | | |
| DESCRIPTION | <p>new_item() creates a new item from <i>name</i> and <i>description</i>, and returns a pointer to the new item.</p> <p>free_item() frees the storage allocated for <i>item</i>. Once an item is freed, the user can no longer connect it to a menu.</p> | | | | | | | | |
| RETURN VALUES | <p>new_item() returns NULL on error.</p> <p>free_item() returns one of the following:</p> <table border="0"> <tr> <td>E_OK</td> <td>The routine returned successfully.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> <tr> <td>E_BAD_ARGUMENT</td> <td>An incorrect argument was passed to the routine.</td> </tr> <tr> <td>E_CONNECTED</td> <td>One or more items are already connected to another menu.</td> </tr> </table> | E_OK | The routine returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An incorrect argument was passed to the routine. | E_CONNECTED | One or more items are already connected to another menu. |
| E_OK | The routine returned successfully. | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. | | | | | | | | |
| E_CONNECTED | One or more items are already connected to another menu. | | | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | | | |
| | <table border="1"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | | | |
| MT-Level | Unsafe | | | | | | | | |
| SEE ALSO | curses(3CURSES), menus(3CURSES), attributes(5) | | | | | | | | |
| NOTES | The header <menu.h> automatically includes the headers <eti.h> and <curses.h>. | | | | | | | | |

new_menu(3CURSES)

NAME menu_new, new_menu, free_menu – create and destroy menus

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

MENU *new_menu (ITEM **items) ;
int free_menu (MENU *menu) ;
```

DESCRIPTION new_menu () creates a new menu connected to the item pointer array *items* and returns a pointer to the new menu.

free_menu () disconnects *menu* from its associated item pointer array and frees the storage allocated for the menu.

RETURN VALUES new_menu () returns NULL on error.

free_menu () returns one of the following:

| | |
|----------------|--|
| E_OK | The routine returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. |
| E_POSTED | The menu is already posted. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), attributes(5)

NOTES The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

| | |
|--------------------|--|
| NAME | curs_pad, newpad, subpad, prefresh, pnoutrefresh, pechochar, pechowchar – create and display curses pads |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> WINDOW *newpad(int nlines, int ncols); WINDOW *subpad(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); int prefresh(WINDOW *pad, int pminrow, int pmincol, int sminrow, int smincol, int smaxrow, int smaxcol); int pnoutrefresh(WINDOW *pad, int pminrow, int pmincol, int sminrow, int smincol, int smaxrow, int smaxcol); int pechochar(WINDOW *pad, chtype ch); int pechowchar(WINDOW *pad, chtype wch);</pre> |
| DESCRIPTION | <p>The <code>newpad()</code> routine creates and returns a pointer to a new pad data structure with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. A pad is like a window, except that it is not restricted by the screen size, and is not necessarily associated with a particular part of the screen. Pads can be used when a large window is needed, and only a part of the window will be on the screen at one time. Automatic refreshes of pads (for example, from scrolling or echoing of input) do not occur. It is not legal to call <code>wrefresh(3CURSES)</code> with a <i>pad</i> as an argument; the routines <code>prefresh()</code> or <code>pnoutrefresh()</code> should be called instead. Note that these routines require additional parameters to specify the part of the pad to be displayed and the location on the screen to be used for the display.</p> <p>The <code>subpad()</code> routine creates and returns a pointer to a subwindow within a pad with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. Unlike <code>subwin(3CURSES)</code>, which uses screen coordinates, the window is at position (<i>begin_x</i>, <i>begin_y</i>) on the pad. The window is made in the middle of the window <i>orig</i>, so that changes made to one window affect both windows. During the use of this routine, it will often be necessary to call <code>touchwin(3CURSES)</code> or <code>touchline(3CURSES)</code> on <i>orig</i> before calling <code>prefresh()</code>.</p> <p>The <code>prefresh()</code> and <code>pnoutrefresh()</code> routines are analogous to <code>wrefresh(3CURSES)</code> and <code>wnoutrefresh(3CURSES)</code> except that they relate to pads instead of windows. The additional parameters are needed to indicate what part of the pad and screen are involved. <i>pminrow</i> and <i>pmincol</i> specify the upper left-hand corner of the rectangle to be displayed in the pad. <i>sminrow</i>, <i>smincol</i>, <i>smaxrow</i>, and <i>smaxcol</i> specify the edges of the rectangle to be displayed on the screen. The lower right-hand corner of the rectangle to be displayed in the pad is calculated from the screen coordinates, since the rectangles must be the same size. Both rectangles must be entirely contained within their respective structures. Negative values of <i>pminrow</i>, <i>pmincol</i>, <i>sminrow</i>, or <i>smincol</i> are treated as if they were zero.</p> |

newpad(3CURSES)

The `pechochar()` routine is functionally equivalent to a call to `addch(3CURSES)` followed by a call to `refresh(3CURSES)`, a call to `waddch(3CURSES)` followed by a call to `wrefresh(3CURSES)`, or a call to `waddch(3CURSES)` followed by a call to `prefresh()`. The knowledge that only a single character is being output is taken into consideration and, for non-control characters, a considerable performance gain might be seen by using these routines instead of their equivalents. In the case of `pechochar()`, the last location of the pad on the screen is reused for the arguments to `prefresh()`.

RETURN VALUES Routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

Routines that return pointers return `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `addch(3CURSES)`, `curses(3CURSES)`, `refresh(3CURSES)`, `subwin(3CURSES)`, `touchline(3CURSES)`, `touchwin(3CURSES)`, `waddch(3CURSES)`, `wnoutrefresh(3CURSES)`, `wrefresh(3CURSES)`, `attributes(5)`

NOTES The header file `<curses.h>` automatically includes the header files `<stdio.h>`, `<unctrl.h>` and `<widec.h>`.

Note that `pechochar()` may be a macro.

| | | | | | | | | | | | | | | | | | | | | | |
|--------------------|---|---------------|--|--------------|--|------------|-------------------------------------|----------------|--|----------------|--|----------------|---|----------------|--|----------------|--|----------------|---|-------------|---|
| NAME | newpad, pnoutrefresh, prefresh, subpad – create or refresh a pad or subpad | | | | | | | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> WINDOW *newpad(int nlines, int ncols); int pnoutrefresh(WINDOW *pad, int pminrow, int pmincol, int sminrow, int smincol, int smaxrow, int smaxcol); int prefresh(WINDOW *pad, int pminrow, int pmincol, int sminrow, int smincol, int smaxrow, int smaxcol); WINDOW *subpad(WINDOW *orig, int nlines, int ncols);</pre> | | | | | | | | | | | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="vertical-align: top;"><i>nlines</i></td> <td>Is the number of lines in the pad to be created.</td> </tr> <tr> <td style="vertical-align: top;"><i>ncols</i></td> <td>Is the number of columns in the pad to be created.</td> </tr> <tr> <td style="vertical-align: top;"><i>pad</i></td> <td>Is a pointer to the pad to refresh.</td> </tr> <tr> <td style="vertical-align: top;"><i>pminrow</i></td> <td>Is the row coordinate of the upper left corner of the pad rectangle to be copied</td> </tr> <tr> <td style="vertical-align: top;"><i>pmincol</i></td> <td>Is the column coordinate of the upper left corner of the pad rectangle to be copied.</td> </tr> <tr> <td style="vertical-align: top;"><i>sminrow</i></td> <td>Is the row coordinate of the upper left corner of the rectangle on the physical screen where pad is to be positioned.</td> </tr> <tr> <td style="vertical-align: top;"><i>smincol</i></td> <td>Is the column coordinate of the upper left corner of the rectangle on the physical screen where pad is to be positioned.</td> </tr> <tr> <td style="vertical-align: top;"><i>smaxrow</i></td> <td>Is the row coordinate of the lower right corner of the rectangle on the physical screen where the pad is to be positioned.</td> </tr> <tr> <td style="vertical-align: top;"><i>smaxcol</i></td> <td>Is the column coordinate of the lower right corner of the rectangle on the physical screen where the pad is to be positioned.</td> </tr> <tr> <td style="vertical-align: top;"><i>orig</i></td> <td>Is a pointer to the parent pad within which a sub-pad is created.</td> </tr> </table> | <i>nlines</i> | Is the number of lines in the pad to be created. | <i>ncols</i> | Is the number of columns in the pad to be created. | <i>pad</i> | Is a pointer to the pad to refresh. | <i>pminrow</i> | Is the row coordinate of the upper left corner of the pad rectangle to be copied | <i>pmincol</i> | Is the column coordinate of the upper left corner of the pad rectangle to be copied. | <i>sminrow</i> | Is the row coordinate of the upper left corner of the rectangle on the physical screen where pad is to be positioned. | <i>smincol</i> | Is the column coordinate of the upper left corner of the rectangle on the physical screen where pad is to be positioned. | <i>smaxrow</i> | Is the row coordinate of the lower right corner of the rectangle on the physical screen where the pad is to be positioned. | <i>smaxcol</i> | Is the column coordinate of the lower right corner of the rectangle on the physical screen where the pad is to be positioned. | <i>orig</i> | Is a pointer to the parent pad within which a sub-pad is created. |
| <i>nlines</i> | Is the number of lines in the pad to be created. | | | | | | | | | | | | | | | | | | | | |
| <i>ncols</i> | Is the number of columns in the pad to be created. | | | | | | | | | | | | | | | | | | | | |
| <i>pad</i> | Is a pointer to the pad to refresh. | | | | | | | | | | | | | | | | | | | | |
| <i>pminrow</i> | Is the row coordinate of the upper left corner of the pad rectangle to be copied | | | | | | | | | | | | | | | | | | | | |
| <i>pmincol</i> | Is the column coordinate of the upper left corner of the pad rectangle to be copied. | | | | | | | | | | | | | | | | | | | | |
| <i>sminrow</i> | Is the row coordinate of the upper left corner of the rectangle on the physical screen where pad is to be positioned. | | | | | | | | | | | | | | | | | | | | |
| <i>smincol</i> | Is the column coordinate of the upper left corner of the rectangle on the physical screen where pad is to be positioned. | | | | | | | | | | | | | | | | | | | | |
| <i>smaxrow</i> | Is the row coordinate of the lower right corner of the rectangle on the physical screen where the pad is to be positioned. | | | | | | | | | | | | | | | | | | | | |
| <i>smaxcol</i> | Is the column coordinate of the lower right corner of the rectangle on the physical screen where the pad is to be positioned. | | | | | | | | | | | | | | | | | | | | |
| <i>orig</i> | Is a pointer to the parent pad within which a sub-pad is created. | | | | | | | | | | | | | | | | | | | | |
| DESCRIPTION | The newpad() function creates a new pad with the specified number of lines and columns. A pointer to the new pad structure is returned. A pad differs from a window in that it is not restricted to the size of the physical screen. It is useful when only part of a large window will be displayed at any one time. | | | | | | | | | | | | | | | | | | | | |

newpad(3XCURSES)

Automatic refreshes by scrolling or echoing of input do not take place when pads are used. Pads have their own refresh commands, `prefresh()` and `pnoutrefresh()`.

The `prefresh()` function copies the specified portion of the logical pad to the terminal screen. The parameters `pmincol` and `pminrow` specify the upper left corner of the rectangular area of the pad to be displayed. The lower right coordinate of the rectangular area of the pad that is to be displayed is calculated from the screen parameters (`sminrow`, `smincol`, `smaxrow`, `smaxcol`).

This function calls the `pnoutrefresh()` function to copy the specified portion of *pad* to the terminal screen and the `doupdate(3XCURSES)` function to do the actual update. The logical cursor is copied to the same location in the physical window unless `leaveok(3XCURSES)` is enabled (in which case, the cursor is placed in a position that the program finds convenient).

When outputting several pads at once, it is often more efficient to call the `pnoutrefresh()` and `doupdate()` functions directly. A call to `pnoutrefresh()` for each pad first, followed by only one call to `doupdate()` to update the screen, results in one burst of output, fewer characters sent, and less CPU time used.

The `subpad()` function creates a sub-pad within the pad *orig* with the specified number of lines and columns. A pointer to the new pad structure is returned. The sub-pad is positioned in the middle of *orig*. Any changes made to one pad affect the other. `touchwin(3XCURSES)` or `touchline(3XCURSES)` will likely have to be called on pad *orig* to correctly update the window.

RETURN VALUES On success, the `newpad()` and `subpad()` functions returns a pointer to the new pad data structure. Otherwise, they return a null pointer.

On success, the `pnoutrefresh()` and `prefresh()` functions return `OK`. Otherwise, they return `ERR`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `clearok(3XCURSES)`, `doupdate(3XCURSES)`, `is_linetouched(3XCURSES)`, `libcurses(3XCURSES)`, `pechochar(3XCURSES)`, `attributes(5)`, `standards(5)`

NAME form_new_page, set_new_page, new_page – forms pagination

SYNOPSIS

```
cc [ flag... ] file... -lform -lcurses [ library... ]
#include <form.h>

int set_new_page(FIELD *field, int bool);
int new_page(FIELD *field);
```

DESCRIPTION set_new_page() marks *field* as the beginning of a new page on the form.

new_page() returns a boolean value indicating whether or not *field* begins a new page of the form.

RETURN VALUES new_page returns TRUE or FALSE.

set_new_page() returns one of the following:

| | |
|----------------|---|
| E_OK | The function returned successfully. |
| E_CONNECTED | The field is already connected to a form. |
| E_SYSTEM_ERROR | System error. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), forms(3CURSES), attributes(5)

NOTES The header <form.h> automatically includes the headers <eti.h> and <curses.h>.

new_panel(3CURSES)

NAME panel_new, new_panel, del_panel – create and destroy panels

SYNOPSIS

```
cc [ flag ... ] file ... -lpanel -lcurses [ library .. ]
#include <panel.h>

PANEL *new_panel (WINDOW *win) ;
int del_panel (PANEL *panel) ;
```

DESCRIPTION new_panel () creates a new panel associated with *win* and returns the panel pointer. The new panel is placed on top of the panel deck.

del_panel () destroys *panel*, but not its associated window.

RETURN VALUES new_panel () returns NULL if an error occurs.

del_win () returns OK if successful, ERR otherwise.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), panel_update(3CURSES), panels(3CURSES), attributes(5)

NOTES The header <panel.h> automatically includes the header <curses.h>.

| | |
|--------------------|--|
| NAME | curs_initscr, initscr, newterm, endwin, isendwin, set_term, delscreen – curses screen initialization and manipulation routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> WINDOW *initscr(void); int endwin(void); int isendwin(void); SCREEN *newterm(char *<i>type</i>, FILE *<i>outfd</i>, FILE *<i>infd</i>); SCREEN *set_term(SCREEN *<i>new</i>); void delscreen(SCREEN * <i>sp</i>);</pre> |
| DESCRIPTION | <p><code>initscr()</code> is almost always the first routine that should be called (the exceptions are <code>slk_init()</code>, <code>filter()</code>, <code>ripoffline()</code>, <code>use_env()</code> and, for multiple-terminal applications, <code>newterm()</code>.) This determines the terminal type and initializes all curses data structures. <code>initscr()</code> also causes the first call to <code>refresh()</code> to clear the screen. If errors occur, <code>initscr()</code> writes an appropriate error message to standard error and exits; otherwise, a pointer is returned to <code>stdscr()</code>. If the program needs an indication of error conditions, <code>newterm()</code> should be used instead of <code>initscr()</code>; <code>initscr()</code> should only be called once per application.</p> <p>A program that outputs to more than one terminal should use the <code>newterm()</code> routine for each terminal instead of <code>initscr()</code>. A program that needs an indication of error conditions, so it can continue to run in a line-oriented mode if the terminal cannot support a screen-oriented program, would also use this routine. The routine <code>newterm()</code> should be called once for each terminal. It returns a variable of type <code>SCREEN *</code> which should be saved as a reference to that terminal. The arguments are the <i>type</i> of the terminal to be used in place of <code>\$TERM</code>, a file pointer for output to the terminal, and another file pointer for input from the terminal (if <i>type</i> is <code>NULL</code>, <code>\$TERM</code> will be used). The program must also call <code>endwin()</code> for each terminal being used before exiting from curses. If <code>newterm()</code> is called more than once for the same terminal, the first terminal referred to must be the last one for which <code>endwin()</code> is called.</p> <p>A program should always call <code>endwin()</code> before exiting or escaping from curses mode temporarily. This routine restores tty modes, moves the cursor to the lower left-hand corner of the screen and resets the terminal into the proper non-visual mode. Calling <code>refresh()</code> or <code>doupdate()</code> after a temporary escape causes the program to resume visual mode.</p> <p>The <code>isendwin()</code> routine returns <code>TRUE</code> if <code>endwin()</code> has been called without any subsequent calls to <code>wrefresh()</code>, and <code>FALSE</code> otherwise.</p> |

newterm(3CURSES)

The `set_term()` routine is used to switch between different terminals. The screen reference `new` becomes the new current terminal. The previous terminal is returned by the routine. This is the only routine which manipulates `SCREEN` pointers; all other routines affect only the current terminal.

The `delscreen()` routine frees storage associated with the `SCREEN` data structure. The `endwin()` routine does not do this, so `delscreen()` should be called after `endwin()` if a particular `SCREEN` is no longer needed.

RETURN VALUES `endwin()` returns the integer `ERR` upon failure and `OK` upon successful completion.

Routines that return pointers always return `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_kernel(3CURSES)`, `curl_refresh(3CURSES)`, `curl_slk(3CURSES)`, `curl_util(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `initscr()` and `newterm()` may be macros.

| | |
|----------------------|---|
| NAME | initscr, newterm – screen initialization functions |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> WINDOW *initscr(void); SCREEN *newterm(char *type, FILE *outfp, FILE *infp);</pre> |
| PARAMETERS | <p><i>type</i> Is a string defining the terminal type to be used in place of TERM.</p> <p><i>outfp</i> Is a pointer to a file to be used for output to the terminal.</p> <p><i>infp</i> Is the pointer to a file to be used for input to the terminal.</p> |
| DESCRIPTION | <p>The <code>initscr()</code> function initializes X/Open Curses data structures, determines the terminal type, and ensures the first call to <code>refresh(3XCURSES)</code> clears the screen.</p> <p>The <code>newterm()</code> function opens a new terminal with each call. It should be used instead of <code>initscr()</code> when the program interacts with more than one terminal. It returns a variable of type <code>SCREEN</code>, which should be used for later reference to that terminal. Before program termination, <code>endwin()</code> should be called for each terminal.</p> <p>The only functions that you can call before calling <code>initscr()</code> or <code>newterm()</code> are <code>filter(3XCURSES)</code>, <code>riporffline(3XCURSES)</code>, <code>slk_init(3XCURSES)</code>, and <code>use_env(3XCURSES)</code>.</p> |
| RETURN VALUES | <p>On success, the <code>initscr()</code> function returns a pointer to <code>stdscr</code>; otherwise, <code>initscr()</code> does not return.</p> <p>On success, the <code>newterm()</code> function returns a pointer to the specified terminal; otherwise, a null pointer is returned.</p> |
| ERRORS | None. |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: |

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

`newterm(3XCURSES)`

SEE ALSO | `del_curterm(3XCURSES)`, `delscreen(3XCURSES)`, `doupdate(3XCURSES)`,
`endwin(3XCURSES)`, `filter(3XCURSES)`, `libcurses(3XCURSES)`,
`slk_attroff(3XCURSES)`, `use_env(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | curs_window, newwin, delwin, mvwin, subwin, derwin, mvderwin, dupwin, wsyncup, syncok, wcursyncup, wsyncdown – create curses windows |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> WINDOW *newwin(int nlines, int ncols, int begin_y, int begin_x); int delwin(WINDOW *win); int mvwin(WINDOW *win, int y, int x); WINDOW *subwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); WINDOW *derwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); int mvderwin(WINDOW *win, int par_y, int par_x); WINDOW *dupwin(WINDOW *win); void wsyncup(WINDOW *win); int syncok(WINDOW *win, bool bf); void wcursyncup(WINDOW *win); void wsyncdown(WINDOW *win);</pre> |
| DESCRIPTION | <p>The <code>newwin()</code> routine creates and returns a pointer to a new window with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. The upper left-hand corner of the window is at line <i>begin_y</i>, column <i>begin_x</i>. If either <i>nlines</i> or <i>ncols</i> is zero, they default to LINES — <i>begin_y</i> and COLS — <i>begin_x</i>. A new full-screen window is created by calling <code>newwin(0, 0, 0, 0)</code>.</p> <p>The <code>delwin()</code> routine deletes the named window, freeing all memory associated with it. Subwindows must be deleted before the main window can be deleted.</p> <p>The <code>mvwin()</code> routine moves the window so that the upper left-hand corner is at position (<i>x</i>, <i>y</i>). If the move would cause the window to be off the screen, it is an error and the window is not moved. Moving subwindows is allowed, but should be avoided.</p> <p>The <code>subwin()</code> routine creates and returns a pointer to a new window with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. The window is at position (<i>begin_y</i>, <i>begin_x</i>) on the screen. (This position is relative to the screen, and not to the window <i>orig</i>.) The window is made in the middle of the window <i>orig</i>, so that changes made to one window will affect both windows. The subwindow shares memory with the window <i>orig</i>. When using this routine, it is necessary to call <code>touchwin()</code> or <code>touchline()</code> on <i>orig</i> before calling <code>wrefresh()</code> on the subwindow.</p> <p>The <code>derwin()</code> routine is the same as <code>subwin()</code>, except that <i>begin_y</i> and <i>begin_x</i> are relative to the origin of the window <i>orig</i> rather than the screen. There is no difference between the subwindows and the derived windows.</p> |

newwin(3CURSES)

The `mvderwin()` routine moves a derived window (or subwindow) inside its parent window. The screen-relative parameters of the window are not changed. This routine is used to display different parts of the parent window at the same physical position on the screen.

The `dupwin()` routine creates an exact duplicate of the window *win*.

Each `curses` window maintains two data structures: the character image structure and the status structure. The character image structure is shared among all windows in the window hierarchy (that is, the window with all subwindows). The status structure, which contains information about individual line changes in the window, is private to each window. The routine `wrefresh()` uses the status data structure when performing screen updating. Since status structures are not shared, changes made to one window in the hierarchy may not be properly reflected on the screen.

The routine `wsyncup()` causes the changes in the status structure of a window to be reflected in the status structures of its ancestors. If `syncok()` is called with second argument `TRUE` then `wsyncup()` is called automatically whenever there is a change in the window.

The routine `wcursyncup()` updates the current cursor position of all the ancestors of the window to reflect the current cursor position of the window.

The routine `wsyncdown()` updates the status structure of the window to reflect the changes in the status structures of its ancestors. Applications seldom call this routine because it is called automatically by `wrefresh()`.

RETURN VALUES

Routines that return an integer return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`delwin()` returns the integer `ERR` upon failure and `OK` upon successful completion.

Routines that return pointers return `NULL` on error.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_refresh(3CURSES)`, `curs_touch(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

If many small changes are made to the window, the `wsyncup()` option could degrade performance.

`newwin(3CURSES)`

Note that `syncok()` may be a macro.

newwin(3XCURSES)

| | | | | | | | | | |
|--------------------|--|-------------|--|---------------|--|--------------|--|----------------|---|
| NAME | derwin, newwin, subwin – create a new window or subwindow | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> WINDOW *derwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); WINDOW *newwin(int nlines, int ncols, int begin_y, int begin_x); WINDOW *subwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x);</pre> | | | | | | | | |
| DESCRIPTION | <p>The <code>derwin()</code> function creates a subwindow within window <i>orig</i>, with the specified number of lines and columns, and upper left corner positioned at <i>begin_x</i>, <i>begin_y</i> relative to window <i>orig</i>. A pointer to the new window structure is returned.</p> <p>The <code>newwin()</code> function creates a new window with the specified number of lines and columns and upper left corner positioned at <i>begin_x</i>, <i>begin_y</i>. A pointer to the new window structure is returned. A full-screen window can be created by calling <code>newwin(0, 0, 0, 0)</code>.</p> <p>If the number of lines specified is zero, <code>newwin()</code> uses a default value of <code>LINES</code> minus <i>begin_y</i>; if the number of columns specified is zero, <code>newwin()</code> uses the default value of <code>COLS</code> minus <i>begin_x</i>.</p> <p>The <code>subwin()</code> function creates a subwindow within window <i>orig</i>, with the specified number of lines and columns, and upper left corner positioned at <i>begin_x</i>, <i>begin_y</i> (relative to the physical screen, <i>not</i> to window <i>orig</i>). A pointer to the new window structure is returned.</p> <p>The original window and subwindow share character storage of the overlapping area (each window maintains its own pointers, cursor location, and other items). This means that characters and attributes are identical in overlapping areas regardless of which window characters are written to.</p> <p>When using subwindows, it is often necessary to call <code>touchwin(3XCURSES)</code> before <code>wrefresh(3XCURSES)</code> to maintain proper screen contents.</p> | | | | | | | | |
| PARAMETERS | <table><tr><td><i>orig</i></td><td>Is a pointer to the parent window for the newly created subwindow.</td></tr><tr><td><i>nlines</i></td><td>Is the number of lines in the subwindow.</td></tr><tr><td><i>ncols</i></td><td>Is the number of columns in the subwindow.</td></tr><tr><td><i>begin_y</i></td><td>Is the y (row) coordinate of the upper left corner of the subwindow, relative to the parent window.</td></tr></table> | <i>orig</i> | Is a pointer to the parent window for the newly created subwindow. | <i>nlines</i> | Is the number of lines in the subwindow. | <i>ncols</i> | Is the number of columns in the subwindow. | <i>begin_y</i> | Is the y (row) coordinate of the upper left corner of the subwindow, relative to the parent window. |
| <i>orig</i> | Is a pointer to the parent window for the newly created subwindow. | | | | | | | | |
| <i>nlines</i> | Is the number of lines in the subwindow. | | | | | | | | |
| <i>ncols</i> | Is the number of columns in the subwindow. | | | | | | | | |
| <i>begin_y</i> | Is the y (row) coordinate of the upper left corner of the subwindow, relative to the parent window. | | | | | | | | |

`newwin(3XCURSES)`

begin_x Is the x (column) coordinate of the upper left corner of the subwindow, relative to the parent window.

RETURN VALUES On success, these functions return a pointer to the newly-created window. Otherwise, they return `ERR`.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `douupdate(3XCURSES)`, `is_linetouched(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

nl(3CURSES)

| | |
|--------------------|--|
| NAME | <code> curs_ouptos, clearok, idlok, idcok, immedok, leaveok, setscreg, wsetscreg, scrollok, nl, nonl</code> – curses terminal output option control routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int clearok(WINDOW *win, bool bf); int idlok(WINDOW *win, bool bf); void idcok(WINDOW *win, bool bf); void immedok(WINDOW *win, bool bf); int leaveok(WINDOW *win, bool bf); int setscreg(int top, int bot); int wsetscreg(WINDOW *win, int top, int bot); int scrollok(WINDOW *win, bool bf); int nl(void); int nonl(void);</pre> |
| DESCRIPTION | <p>These routines set options that deal with output within curses. All options are initially FALSE, unless otherwise stated. It is not necessary to turn these options off before calling <code>endwin()</code>.</p> <p>With the <code>clearok()</code> routine, if enabled (<i>bf</i> is TRUE), the next call to <code>wrefresh()</code> with this window will clear the screen completely and redraw the entire screen from scratch. This is useful when the contents of the screen are uncertain, or in some cases for a more pleasing visual effect. If the <i>win</i> argument to <code>clearok()</code> is the global variable <code>curscr()</code>, the next call to <code>wrefresh()</code> with any window causes the screen to be cleared and repainted from scratch.</p> <p>With the <code>idlok()</code> routine, if enabled (<i>bf</i> is TRUE), curses considers using the hardware insert/delete line feature of terminals so equipped. If disabled (<i>bf</i> is FALSE), curses very seldom uses this feature. (The insert/delete character feature is always considered.) This option should be enabled only if the application needs insert/delete line, for example, for a screen editor. It is disabled by default because insert/delete line tends to be visually annoying when used in applications where it isn't really needed. If insert/delete line cannot be used, curses redraws the changed portions of all lines.</p> <p>With the <code>idcok()</code> routine, if enabled (<i>bf</i> is TRUE), curses considers using the hardware insert/delete character feature of terminals so equipped. This is enabled by default.</p> <p>With the <code>immedok()</code> routine, if enabled (<i>bf</i> is TRUE), any change in the window image, such as the ones caused by <code>waddch()</code>, <code>wclrtoeol()</code>, <code>wscrl()</code>, etc., automatically cause a call to <code>wrefresh()</code>. However, it may degrade the performance considerably, due to repeated calls to <code>wrefresh()</code>. It is disabled by default.</p> |

nl(3CURSES)

Normally, the hardware cursor is left at the location of the window cursor being refreshed. The `leaveok()` option allows the cursor to be left wherever the update happens to leave it. It is useful for applications where the cursor is not used, since it reduces the need for cursor motions. If possible, the cursor is made invisible when this option is enabled.

The `setscrreg()` and `wsetscrreg()` routines allow the application programmer to set a software scrolling region in a window. *top* and *bot* are the line numbers of the top and bottom margin of the scrolling region. (Line 0 is the top line of the window.) If this option and `scrollok()` are enabled, an attempt to move off the bottom margin line causes all lines in the scrolling region to scroll up one line. Only the text of the window is scrolled. (Note that this has nothing to do with the use of a physical scrolling region capability in the terminal, like that in the VT100. If `idlok()` is enabled and the terminal has either a scrolling region or insert/delete line capability, they will probably be used by the output routines.)

The `scrollok()` option controls what happens when the cursor of a window is moved off the edge of the window or scrolling region, either as a result of a newline action on the bottom line, or typing the last character of the last line. If disabled, (*bf* is FALSE), the cursor is left on the bottom line. If enabled, (*bf* is TRUE), `wrefresh()` is called on the window, and the physical terminal and window are scrolled up one line. (Note that in order to get the physical scrolling effect on the terminal, it is also necessary to call `idlok()`.)

The `nl()` and `nonl()` routines control whether newline is translated into carriage return and linefeed on output, and whether return is translated into newline on input. Initially, the translations do occur. By disabling these translations using `nonl()`, `curses` is able to make better use of the linefeed capability, resulting in faster cursor motion.

RETURN VALUES `setscrreg()` and `wsetscrreg()` return OK upon success and ERR upon failure. All other routines that return an integer always return OK.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_addch(3CURSES)`, `curs_clear(3CURSES)`, `curs_initscr(3CURSES)`, `curs_refresh(3CURSES)`, `curs_scroll(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `clearok()`, `leaveok()`, `scrollok()`, `idcok()`, `nl()`, `nonl()`, and `setscrreg()` may be macros.

nl(3CURSES)

The `immedok()` routine is useful for windows that are used as terminal emulators.

NAME nl, nonl – enable/disable newline control

SYNOPSIS `cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \`
`-R /usr/xpg4/lib -lcurses [library...]`

`c89 [flag...] file... -lcurses [library...]`

```
#include <curses.h>
int nl(void);
int nonl(void);
```

DESCRIPTION The nl () function enables the handling of newlines. The nl () function converts newline into carriage return and line feed on output and converts carriage return into newline on input. nonl () disables the handling of newlines.

The handling of newlines is initially enabled. Disabling the handling of newlines results in faster cursor motion since X/Open Curses can use the line-feed capability more efficiently.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO libcurses(3XCURSES), attributes(5), standards(5)

nocbreak(3CURSES)

| | |
|--------------------|---|
| NAME | <code>curs_inopts</code> , <code>cbreak</code> , <code>nocbreak</code> , <code>echo</code> , <code>noecho</code> , <code>halfdelay</code> , <code>intrflush</code> , <code>keypad</code> , <code>meta</code> , <code>nodelay</code> , <code>notimeout</code> , <code>raw</code> , <code>noraw</code> , <code>noqiflush</code> , <code>qiflush</code> , <code>timeout</code> , <code>wtimeout</code> , <code>typeahead</code> – curses terminal input option control routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int cbreak(void); int nocbreak(void); int echo(void); int noecho(void); int halfdelay(int <i>tenths</i>); int intrflush(WINDOW *<i>win</i>, bool <i>bf</i>); int keypad(WINDOW *<i>win</i>, bool <i>bf</i>); int meta(WINDOW *<i>win</i>, bool <i>bf</i>); int nodelay(WINDOW *<i>win</i>, bool <i>bf</i>); int notimeout(WINDOW *<i>win</i>, bool <i>bf</i>); int raw(void); int noraw(void); void noqiflush(void); void qiflush(void); void timeout(int <i>delay</i>); void wtimeout(WINDOW *<i>win</i>, int <i>delay</i>); int typeahead(int <i>fildes</i>);</pre> |
| DESCRIPTION | <p>The <code>cbreak()</code> and <code>nocbreak()</code> routines put the terminal into and out of <code>cbreak()</code> mode, respectively. In this mode, characters typed by the user are immediately available to the program, and erase/kill character-processing is not performed. When out of this mode, the tty driver buffers the typed characters until a newline or carriage return is typed. Interrupt and flow control characters are unaffected by this mode. Initially the terminal may or may not be in <code>cbreak()</code> mode, as the mode is inherited; therefore, a program should call <code>cbreak()</code> or <code>nocbreak()</code> explicitly. Most interactive programs using curses set the <code>cbreak()</code> mode.</p> <p>Note that <code>cbreak()</code> overrides <code>raw()</code>. (See <code>curs_getch(3CURSES)</code> for a discussion of how these routines interact with <code>echo()</code> and <code>noecho()</code>.)</p> <p>The <code>echo()</code> and <code>noecho()</code> routines control whether characters typed by the user are echoed by <code>getch()</code> as they are typed. Echoing by the tty driver is always disabled, but initially <code>getch()</code> is in echo mode, so characters typed are echoed. Authors of</p> |

most interactive programs prefer to do their own echoing in a controlled area of the screen, or not to echo at all, so they disable echoing by calling `noecho()`. (See `curs_getch(3CURSES)` for a discussion of how these routines interact with `cbreak()` and `nocbreak()`.)

The `halfdelay()` routine is used for half-delay mode, which is similar to `cbreak()` mode in that characters typed by the user are immediately available to the program. However, after blocking for *tenths* tenths of seconds, `ERR` is returned if nothing has been typed. The value of *tenths* must be a number between 1 and 255. Use `nocbreak()` to leave half-delay mode.

If the `intrflush()` option is enabled, (*bf* is `TRUE`), when an interrupt key is pressed on the keyboard (interrupt, break, quit) all output in the tty driver queue will be flushed, giving the effect of faster response to the interrupt, but causing `curses` to have the wrong idea of what is on the screen. Disabling (*bf* is `FALSE`), the option prevents the flush. The default for the option is inherited from the tty driver settings. The window argument is ignored.

The `keypad()` option enables the keypad of the user's terminal. If enabled (*bf* is `TRUE`), the user can press a function key (such as an arrow key) and `wgetch()` returns a single value representing the function key, as in `KEY_LEFT`. If disabled (*bf* is `FALSE`), `curses` does not treat function keys specially and the program has to interpret the escape sequences itself. If the keypad in the terminal can be turned on (made to transmit) and off (made to work locally), turning on this option causes the terminal keypad to be turned on when `wgetch()` is called. The default value for keypad is false.

Initially, whether the terminal returns 7 or 8 significant bits on input depends on the control mode of the tty driver (see `termio(7I)`). To force 8 bits to be returned, invoke `meta(win, TRUE)`. To force 7 bits to be returned, invoke `meta(win, FALSE)`. The window argument, *win*, is always ignored. If the terminfo capabilities `smm` (`meta_on`) and `rmm` (`meta_off`) are defined for the terminal, `smm` is sent to the terminal when `meta(win, TRUE)` is called and `rmm` is sent when `meta(win, FALSE)` is called.

The `nodelay()` option causes `getch()` to be a non-blocking call. If no input is ready, `getch()` returns `ERR`. If disabled (*bf* is `FALSE`), `getch()` waits until a key is pressed.

While interpreting an input escape sequence, `wgetch()` sets a timer while waiting for the next character. If `notimeout(win, TRUE)` is called, then `wgetch()` does not set a timer. The purpose of the timeout is to differentiate between sequences received from a function key and those typed by a user.

With the `raw()` and `noraw()` routines, the terminal is placed into or out of raw mode. Raw mode is similar to `cbreak()` mode, in that characters typed are immediately passed through to the user program. The differences are that in raw mode, the interrupt, quit, suspend, and flow control characters are all passed through uninterpreted, instead of generating a signal. The behavior of the `BREAK` key depends on other bits in the tty driver that are not set by `curses`.

nocbreak(3CURSES)

When the `noqiflush()` routine is used, normal flush of input and output queues associated with the `INTR`, `QUIT` and `SUSP` characters will not be done (see `termio(7I)`). When `qiflush()` is called, the queues will be flushed when these control characters are read.

The `timeout()` and `wtimeout()` routines set blocking or non-blocking read for a given window. If *delay* is negative, blocking read is used (that is, waits indefinitely for input). If *delay* is zero, then non-blocking read is used (that is, read returns `ERR` if no input is waiting). If *delay* is positive, then read blocks for *delay* milliseconds, and returns `ERR` if there is still no input. Hence, these routines provide the same functionality as `nodelay()`, plus the additional capability of being able to block for only *delay* milliseconds (where *delay* is positive).

`curses` does "line-breakout optimization" by looking for typeahead periodically while updating the screen. If input is found, and it is coming from a tty, the current update is postponed until `refresh()` or `doupdate()` is called again. This allows faster response to commands typed in advance. Normally, the input FILE pointer passed to `newterm()`, or `stdin` in the case that `initscr()` was used, will be used to do this typeahead checking. The `typeahead()` routine specifies that the file descriptor *fildev* is to be used to check for typeahead instead. If *fildev* is `-1`, then no typeahead checking is done.

RETURN VALUES All routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_getch(3CURSES)`, `curs_initscr(3CURSES)`, `curses(3CURSES)`, `attributes(5)`, `termio(7I)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `echo()`, `noecho()`, `halfdelay()`, `intrflush()`, `meta()`, `nodelay()`, `notimeout()`, `noqiflush()`, `qiflush()`, `timeout()`, and `wtimeout()` may be macros.

NAME cbreak, nocbreak, noraw, raw – set input mode controls

SYNOPSIS `cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \`
`-R /usr/xpg4/lib -lcurses [library...]`

`c89 [flag...] file... -lcurses [library...]`

```
#include <curses.h>
int cbreak(void);
int nocbreak(void);
int noraw(void);
int raw(void);
```

DESCRIPTION The `cbreak()` function enables the character input mode. This overrides any previous call to the `raw()` function and turns the `stty` flag `ICANON` off.

The `nocbreak()` function sets the line canonical mode and turns the `stty` flag `ICANON` on without touching the `ISIG` or `IXON` flags.

The `noraw()` function sets the line canonical mode and turns the the `stty` flags `ICANON`, `ISIG`, and `IXON` all on.

The `raw()` function sets the character input mode and turns the `stty` flags `ICANON`, `ISIG`, and `IXON` all off. This mode provides maximum control over input.

It is important to remember that the terminal may or may not be in character mode operation initially. Most interactive programs require `cbreak()` to be enabled.

RETURN VALUES On success, these functions return `OK`. Otherwise, they return `ERR`.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `getch(3XCURSES)`, `halfdelay(3XCURSES)`, `nodelay(3XCURSES)`, `timeout(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`, `termio(7I)`

nodelay(3CURSES)

| | |
|--------------------|---|
| NAME | <code>curs_inopts</code> , <code>cbreak</code> , <code>nocbreak</code> , <code>echo</code> , <code>noecho</code> , <code>halfdelay</code> , <code>intrflush</code> , <code>keypad</code> , <code>meta</code> , <code>nodelay</code> , <code>notimeout</code> , <code>raw</code> , <code>noraw</code> , <code>noqiflush</code> , <code>qiflush</code> , <code>timeout</code> , <code>wtimeout</code> , <code>typeahead</code> – curses terminal input option control routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int cbreak(void); int nocbreak(void); int echo(void); int noecho(void); int halfdelay(int <i>tenths</i>); int intrflush(WINDOW *<i>win</i>, bool <i>bf</i>); int keypad(WINDOW *<i>win</i>, bool <i>bf</i>); int meta(WINDOW *<i>win</i>, bool <i>bf</i>); int nodelay(WINDOW *<i>win</i>, bool <i>bf</i>); int notimeout(WINDOW *<i>win</i>, bool <i>bf</i>); int raw(void); int noraw(void); void noqiflush(void); void qiflush(void); void timeout(int <i>delay</i>); void wtimeout(WINDOW *<i>win</i>, int <i>delay</i>); int typeahead(int <i>fildes</i>);</pre> |
| DESCRIPTION | <p>The <code>cbreak()</code> and <code>nocbreak()</code> routines put the terminal into and out of <code>cbreak()</code> mode, respectively. In this mode, characters typed by the user are immediately available to the program, and erase/kill character-processing is not performed. When out of this mode, the tty driver buffers the typed characters until a newline or carriage return is typed. Interrupt and flow control characters are unaffected by this mode. Initially the terminal may or may not be in <code>cbreak()</code> mode, as the mode is inherited; therefore, a program should call <code>cbreak()</code> or <code>nocbreak()</code> explicitly. Most interactive programs using curses set the <code>cbreak()</code> mode.</p> <p>Note that <code>cbreak()</code> overrides <code>raw()</code>. (See <code>curs_getch(3CURSES)</code> for a discussion of how these routines interact with <code>echo()</code> and <code>noecho()</code>.)</p> <p>The <code>echo()</code> and <code>noecho()</code> routines control whether characters typed by the user are echoed by <code>getch()</code> as they are typed. Echoing by the tty driver is always disabled, but initially <code>getch()</code> is in echo mode, so characters typed are echoed. Authors of</p> |

most interactive programs prefer to do their own echoing in a controlled area of the screen, or not to echo at all, so they disable echoing by calling `noecho()`. (See `curs_getch(3CURSES)` for a discussion of how these routines interact with `cbreak()` and `nocbreak()`.)

The `halfdelay()` routine is used for half-delay mode, which is similar to `cbreak()` mode in that characters typed by the user are immediately available to the program. However, after blocking for *tenths* tenths of seconds, `ERR` is returned if nothing has been typed. The value of *tenths* must be a number between 1 and 255. Use `nocbreak()` to leave half-delay mode.

If the `intrflush()` option is enabled, (*bf* is `TRUE`), when an interrupt key is pressed on the keyboard (interrupt, break, quit) all output in the tty driver queue will be flushed, giving the effect of faster response to the interrupt, but causing `curses` to have the wrong idea of what is on the screen. Disabling (*bf* is `FALSE`), the option prevents the flush. The default for the option is inherited from the tty driver settings. The window argument is ignored.

The `keypad()` option enables the keypad of the user's terminal. If enabled (*bf* is `TRUE`), the user can press a function key (such as an arrow key) and `wgetch()` returns a single value representing the function key, as in `KEY_LEFT`. If disabled (*bf* is `FALSE`), `curses` does not treat function keys specially and the program has to interpret the escape sequences itself. If the keypad in the terminal can be turned on (made to transmit) and off (made to work locally), turning on this option causes the terminal keypad to be turned on when `wgetch()` is called. The default value for keypad is false.

Initially, whether the terminal returns 7 or 8 significant bits on input depends on the control mode of the tty driver (see `termio(7I)`). To force 8 bits to be returned, invoke `meta(win, TRUE)`. To force 7 bits to be returned, invoke `meta(win, FALSE)`. The window argument, *win*, is always ignored. If the terminfo capabilities `smm` (`meta_on`) and `rmm` (`meta_off`) are defined for the terminal, `smm` is sent to the terminal when `meta(win, TRUE)` is called and `rmm` is sent when `meta(win, FALSE)` is called.

The `nodelay()` option causes `getch()` to be a non-blocking call. If no input is ready, `getch()` returns `ERR`. If disabled (*bf* is `FALSE`), `getch()` waits until a key is pressed.

While interpreting an input escape sequence, `wgetch()` sets a timer while waiting for the next character. If `notimeout(win, TRUE)` is called, then `wgetch()` does not set a timer. The purpose of the timeout is to differentiate between sequences received from a function key and those typed by a user.

With the `raw()` and `noraw()` routines, the terminal is placed into or out of raw mode. Raw mode is similar to `cbreak()` mode, in that characters typed are immediately passed through to the user program. The differences are that in raw mode, the interrupt, quit, suspend, and flow control characters are all passed through uninterpreted, instead of generating a signal. The behavior of the `BREAK` key depends on other bits in the tty driver that are not set by `curses`.

nodelay(3CURSES)

When the `noqiflush()` routine is used, normal flush of input and output queues associated with the `INTR`, `QUIT` and `SUSP` characters will not be done (see `termio(7I)`). When `qiflush()` is called, the queues will be flushed when these control characters are read.

The `timeout()` and `wtimeout()` routines set blocking or non-blocking read for a given window. If *delay* is negative, blocking read is used (that is, waits indefinitely for input). If *delay* is zero, then non-blocking read is used (that is, read returns `ERR` if no input is waiting). If *delay* is positive, then read blocks for *delay* milliseconds, and returns `ERR` if there is still no input. Hence, these routines provide the same functionality as `nodelay()`, plus the additional capability of being able to block for only *delay* milliseconds (where *delay* is positive).

`curses` does "line-breakout optimization" by looking for typeahead periodically while updating the screen. If input is found, and it is coming from a tty, the current update is postponed until `refresh()` or `doupdate()` is called again. This allows faster response to commands typed in advance. Normally, the input FILE pointer passed to `newterm()`, or `stdin` in the case that `initscr()` was used, will be used to do this typeahead checking. The `typeahead()` routine specifies that the file descriptor *fildev* is to be used to check for typeahead instead. If *fildev* is `-1`, then no typeahead checking is done.

RETURN VALUES

All routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_getch(3CURSES)`, `curs_initscr(3CURSES)`, `curses(3CURSES)`, `attributes(5)`, `termio(7I)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `echo()`, `noecho()`, `halfdelay()`, `intrflush()`, `meta()`, `nodelay()`, `notimeout()`, `noqiflush()`, `qiflush()`, `timeout()`, and `wtimeout()` may be macros.

NAME nodelay – set blocking or non-blocking read

SYNOPSIS `cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \`
`-R /usr/xpg4/lib -lcurses [library...]`

`c89 [flag...] file... -lcurses [library...]`

```
#include <curses.h>

int nodelay(WINDOW *win, bool bf);
```

PARAMETERS *win* Is a pointer to the window in which to enable non-blocking.

bf Is a Boolean expression.

DESCRIPTION If enabled, (*bf* is TRUE), the `nodelay()` function causes `getch(3XCURSES)` to return ERR if no input is ready. When disabled, `getch()` blocks until a key is pressed.

RETURN VALUES On success, the `nodelay()` function returns OK. Otherwise, it returns ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `getch(3XCURSES)`, `halfdelay(3XCURSES)`, `libcurses(3XCURSES)`, `notimeout(3XCURSES)`, `attributes(5)`, `standards(5)`

noecho(3CURSES)

| | |
|--------------------|---|
| NAME | <code>curs_inopts</code> , <code>cbreak</code> , <code>nocbreak</code> , <code>echo</code> , <code>noecho</code> , <code>halfdelay</code> , <code>intrflush</code> , <code>keypad</code> , <code>meta</code> , <code>nodelay</code> , <code>notimeout</code> , <code>raw</code> , <code>noraw</code> , <code>noqiflush</code> , <code>qiflush</code> , <code>timeout</code> , <code>wtimeout</code> , <code>typeahead</code> – curses terminal input option control routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int cbreak(void); int nocbreak(void); int echo(void); int noecho(void); int halfdelay(int <i>tenths</i>); int intrflush(WINDOW *<i>win</i>, bool <i>bf</i>); int keypad(WINDOW *<i>win</i>, bool <i>bf</i>); int meta(WINDOW *<i>win</i>, bool <i>bf</i>); int nodelay(WINDOW *<i>win</i>, bool <i>bf</i>); int notimeout(WINDOW *<i>win</i>, bool <i>bf</i>); int raw(void); int noraw(void); void noqiflush(void); void qiflush(void); void timeout(int <i>delay</i>); void wtimeout(WINDOW *<i>win</i>, int <i>delay</i>); int typeahead(int <i>fildes</i>);</pre> |
| DESCRIPTION | <p>The <code>cbreak()</code> and <code>nocbreak()</code> routines put the terminal into and out of <code>cbreak()</code> mode, respectively. In this mode, characters typed by the user are immediately available to the program, and erase/kill character-processing is not performed. When out of this mode, the tty driver buffers the typed characters until a newline or carriage return is typed. Interrupt and flow control characters are unaffected by this mode. Initially the terminal may or may not be in <code>cbreak()</code> mode, as the mode is inherited; therefore, a program should call <code>cbreak()</code> or <code>nocbreak()</code> explicitly. Most interactive programs using curses set the <code>cbreak()</code> mode.</p> <p>Note that <code>cbreak()</code> overrides <code>raw()</code>. (See <code>curs_getch(3CURSES)</code> for a discussion of how these routines interact with <code>echo()</code> and <code>noecho()</code>.)</p> <p>The <code>echo()</code> and <code>noecho()</code> routines control whether characters typed by the user are echoed by <code>getch()</code> as they are typed. Echoing by the tty driver is always disabled, but initially <code>getch()</code> is in echo mode, so characters typed are echoed. Authors of</p> |

most interactive programs prefer to do their own echoing in a controlled area of the screen, or not to echo at all, so they disable echoing by calling `noecho()`. (See `curs_getch(3CURSES)` for a discussion of how these routines interact with `cbreak()` and `nocbreak()`.)

The `halfdelay()` routine is used for half-delay mode, which is similar to `cbreak()` mode in that characters typed by the user are immediately available to the program. However, after blocking for *tenths* tenths of seconds, `ERR` is returned if nothing has been typed. The value of *tenths* must be a number between 1 and 255. Use `nocbreak()` to leave half-delay mode.

If the `intrflush()` option is enabled, (*bf* is `TRUE`), when an interrupt key is pressed on the keyboard (interrupt, break, quit) all output in the tty driver queue will be flushed, giving the effect of faster response to the interrupt, but causing `curses` to have the wrong idea of what is on the screen. Disabling (*bf* is `FALSE`), the option prevents the flush. The default for the option is inherited from the tty driver settings. The window argument is ignored.

The `keypad()` option enables the keypad of the user's terminal. If enabled (*bf* is `TRUE`), the user can press a function key (such as an arrow key) and `wgetch()` returns a single value representing the function key, as in `KEY_LEFT`. If disabled (*bf* is `FALSE`), `curses` does not treat function keys specially and the program has to interpret the escape sequences itself. If the keypad in the terminal can be turned on (made to transmit) and off (made to work locally), turning on this option causes the terminal keypad to be turned on when `wgetch()` is called. The default value for keypad is false.

Initially, whether the terminal returns 7 or 8 significant bits on input depends on the control mode of the tty driver (see `termio(7I)`). To force 8 bits to be returned, invoke `meta(win, TRUE)`. To force 7 bits to be returned, invoke `meta(win, FALSE)`. The window argument, *win*, is always ignored. If the terminfo capabilities `smm` (`meta_on`) and `rmm` (`meta_off`) are defined for the terminal, `smm` is sent to the terminal when `meta(win, TRUE)` is called and `rmm` is sent when `meta(win, FALSE)` is called.

The `nodelay()` option causes `getch()` to be a non-blocking call. If no input is ready, `getch()` returns `ERR`. If disabled (*bf* is `FALSE`), `getch()` waits until a key is pressed.

While interpreting an input escape sequence, `wgetch()` sets a timer while waiting for the next character. If `notimeout(win, TRUE)` is called, then `wgetch()` does not set a timer. The purpose of the timeout is to differentiate between sequences received from a function key and those typed by a user.

With the `raw()` and `noraw()` routines, the terminal is placed into or out of raw mode. Raw mode is similar to `cbreak()` mode, in that characters typed are immediately passed through to the user program. The differences are that in raw mode, the interrupt, quit, suspend, and flow control characters are all passed through uninterpreted, instead of generating a signal. The behavior of the `BREAK` key depends on other bits in the tty driver that are not set by `curses`.

noecho(3CURSES)

When the `noqiflush()` routine is used, normal flush of input and output queues associated with the `INTR`, `QUIT` and `SUSP` characters will not be done (see `termio(7I)`). When `qiflush()` is called, the queues will be flushed when these control characters are read.

The `timeout()` and `wtimeout()` routines set blocking or non-blocking read for a given window. If *delay* is negative, blocking read is used (that is, waits indefinitely for input). If *delay* is zero, then non-blocking read is used (that is, read returns `ERR` if no input is waiting). If *delay* is positive, then read blocks for *delay* milliseconds, and returns `ERR` if there is still no input. Hence, these routines provide the same functionality as `nodelay()`, plus the additional capability of being able to block for only *delay* milliseconds (where *delay* is positive).

`curses` does "line-breakout optimization" by looking for typeahead periodically while updating the screen. If input is found, and it is coming from a tty, the current update is postponed until `refresh()` or `doupdate()` is called again. This allows faster response to commands typed in advance. Normally, the input FILE pointer passed to `newterm()`, or `stdin` in the case that `initscr()` was used, will be used to do this typeahead checking. The `typeahead()` routine specifies that the file descriptor *fildev* is to be used to check for typeahead instead. If *fildev* is `-1`, then no typeahead checking is done.

RETURN VALUES

All routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_getch(3CURSES)`, `curs_initscr(3CURSES)`, `curses(3CURSES)`, `attributes(5)`, `termio(7I)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `echo()`, `noecho()`, `halfdelay()`, `intrflush()`, `meta()`, `nodelay()`, `notimeout()`, `noqiflush()`, `qiflush()`, `timeout()`, and `wtimeout()` may be macros.

NAME echo, noecho – enable/disable terminal echo

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

int echo(void);

int noecho(void);
```

DESCRIPTION The `echo()` function enables Echo mode for the current screen. The `noecho()` function disables Echo mode for the current screen. Initially, curses software echo mode is enabled and hardware echo mode of the `tty` driver is disabled. The `echo()` and `noecho()` functions control software echo only. Hardware echo must remain disabled for the duration of the application, else the behavior is undefined.

RETURN VALUES Upon successful completion, these functions return OK. Otherwise, they return ERR.

ERRORS No errors are defined.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `getch(3XCURSES)`, `getstr(3XCURSES)`, `initscr(3XCURSES)`, `libcurses(3XCURSES)`, `scanw(3XCURSES)`, `attributes(5)`, `standards(5)`

nonl(3CURSES)

| | |
|--------------------|--|
| NAME | <code> curs_ouptops, clearok, idlok, idcok, immedok, leaveok, setscreg, wsetscreg, scrollok, nl, nonl</code> – curses terminal output option control routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int clearok(WINDOW *win, bool bf); int idlok(WINDOW *win, bool bf); void idcok(WINDOW *win, bool bf); void immedok(WINDOW *win, bool bf); int leaveok(WINDOW *win, bool bf); int setscreg(int top, int bot); int wsetscreg(WINDOW *win, int top, int bot); int scrollok(WINDOW *win, bool bf); int nl(void); int nonl(void);</pre> |
| DESCRIPTION | <p>These routines set options that deal with output within curses. All options are initially FALSE, unless otherwise stated. It is not necessary to turn these options off before calling <code>endwin()</code>.</p> <p>With the <code>clearok()</code> routine, if enabled (<i>bf</i> is TRUE), the next call to <code>wrefresh()</code> with this window will clear the screen completely and redraw the entire screen from scratch. This is useful when the contents of the screen are uncertain, or in some cases for a more pleasing visual effect. If the <i>win</i> argument to <code>clearok()</code> is the global variable <code>curscr()</code>, the next call to <code>wrefresh()</code> with any window causes the screen to be cleared and repainted from scratch.</p> <p>With the <code>idlok()</code> routine, if enabled (<i>bf</i> is TRUE), curses considers using the hardware insert/delete line feature of terminals so equipped. If disabled (<i>bf</i> is FALSE), curses very seldom uses this feature. (The insert/delete character feature is always considered.) This option should be enabled only if the application needs insert/delete line, for example, for a screen editor. It is disabled by default because insert/delete line tends to be visually annoying when used in applications where it isn't really needed. If insert/delete line cannot be used, curses redraws the changed portions of all lines.</p> <p>With the <code>idcok()</code> routine, if enabled (<i>bf</i> is TRUE), curses considers using the hardware insert/delete character feature of terminals so equipped. This is enabled by default.</p> <p>With the <code>immedok()</code> routine, if enabled (<i>bf</i> is TRUE), any change in the window image, such as the ones caused by <code>waddch()</code>, <code>wclrtoeol()</code>, <code>wscrl()</code>, etc., automatically cause a call to <code>wrefresh()</code>. However, it may degrade the performance considerably, due to repeated calls to <code>wrefresh()</code>. It is disabled by default.</p> |

nonl(3CURSES)

Normally, the hardware cursor is left at the location of the window cursor being refreshed. The `leaveok()` option allows the cursor to be left wherever the update happens to leave it. It is useful for applications where the cursor is not used, since it reduces the need for cursor motions. If possible, the cursor is made invisible when this option is enabled.

The `setscrreg()` and `wsetscrreg()` routines allow the application programmer to set a software scrolling region in a window. *top* and *bot* are the line numbers of the top and bottom margin of the scrolling region. (Line 0 is the top line of the window.) If this option and `scrollok()` are enabled, an attempt to move off the bottom margin line causes all lines in the scrolling region to scroll up one line. Only the text of the window is scrolled. (Note that this has nothing to do with the use of a physical scrolling region capability in the terminal, like that in the VT100. If `idlok()` is enabled and the terminal has either a scrolling region or insert/delete line capability, they will probably be used by the output routines.)

The `scrollok()` option controls what happens when the cursor of a window is moved off the edge of the window or scrolling region, either as a result of a newline action on the bottom line, or typing the last character of the last line. If disabled, (*bf* is FALSE), the cursor is left on the bottom line. If enabled, (*bf* is TRUE), `wrefresh()` is called on the window, and the physical terminal and window are scrolled up one line. (Note that in order to get the physical scrolling effect on the terminal, it is also necessary to call `idlok()`.)

The `nl()` and `nonl()` routines control whether newline is translated into carriage return and linefeed on output, and whether return is translated into newline on input. Initially, the translations do occur. By disabling these translations using `nonl()`, `curses` is able to make better use of the linefeed capability, resulting in faster cursor motion.

RETURN VALUES `setscrreg()` and `wsetscrreg()` return OK upon success and ERR upon failure. All other routines that return an integer always return OK.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_addch(3CURSES)`, `curs_clear(3CURSES)`, `curs_initscr(3CURSES)`, `curs_refresh(3CURSES)`, `curs_scroll(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `clearok()`, `leaveok()`, `scrollok()`, `idcok()`, `nl()`, `nonl()`, and `setscrreg()` may be macros.

`nonl(3CURSES)`

The `immedok()` routine is useful for windows that are used as terminal emulators.

| NAME | nl, nonl – enable/disable newline control | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int nl(void); int nonl(void);</pre> | | | | | | |
| DESCRIPTION | <p>The nl () function enables the handling of newlines. The nl () function converts newline into carriage return and line feed on output and converts carriage return into newline on input. nonl () disables the handling of newlines.</p> <p>The handling of newlines is initially enabled. Disabling the handling of newlines results in faster cursor motion since X/Open Curses can use the line-feed capability more efficiently.</p> | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | libcurses(3XCURSES), attributes(5), standards(5) | | | | | | |

noqiflush(3CURSES)

| | |
|--------------------|---|
| NAME | <code>curs_inopts</code> , <code>cbreak</code> , <code>nocbreak</code> , <code>echo</code> , <code>noecho</code> , <code>halfdelay</code> , <code>intrflush</code> , <code>keypad</code> , <code>meta</code> , <code>nodelay</code> , <code>notimeout</code> , <code>raw</code> , <code>noraw</code> , <code>noqiflush</code> , <code>qiflush</code> , <code>timeout</code> , <code>wtimeout</code> , <code>typeahead</code> – curses terminal input option control routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int cbreak(void); int nocbreak(void); int echo(void); int noecho(void); int halfdelay(int <i>tenths</i>); int intrflush(WINDOW *<i>win</i>, bool <i>bf</i>); int keypad(WINDOW *<i>win</i>, bool <i>bf</i>); int meta(WINDOW *<i>win</i>, bool <i>bf</i>); int nodelay(WINDOW *<i>win</i>, bool <i>bf</i>); int notimeout(WINDOW *<i>win</i>, bool <i>bf</i>); int raw(void); int noraw(void); void noqiflush(void); void qiflush(void); void timeout(int <i>delay</i>); void wtimeout(WINDOW *<i>win</i>, int <i>delay</i>); int typeahead(int <i>fildes</i>);</pre> |
| DESCRIPTION | <p>The <code>cbreak()</code> and <code>nocbreak()</code> routines put the terminal into and out of <code>cbreak()</code> mode, respectively. In this mode, characters typed by the user are immediately available to the program, and erase/kill character-processing is not performed. When out of this mode, the tty driver buffers the typed characters until a newline or carriage return is typed. Interrupt and flow control characters are unaffected by this mode. Initially the terminal may or may not be in <code>cbreak()</code> mode, as the mode is inherited; therefore, a program should call <code>cbreak()</code> or <code>nocbreak()</code> explicitly. Most interactive programs using curses set the <code>cbreak()</code> mode.</p> <p>Note that <code>cbreak()</code> overrides <code>raw()</code>. (See <code>curs_getch(3CURSES)</code> for a discussion of how these routines interact with <code>echo()</code> and <code>noecho()</code>.)</p> <p>The <code>echo()</code> and <code>noecho()</code> routines control whether characters typed by the user are echoed by <code>getch()</code> as they are typed. Echoing by the tty driver is always disabled, but initially <code>getch()</code> is in echo mode, so characters typed are echoed. Authors of</p> |

most interactive programs prefer to do their own echoing in a controlled area of the screen, or not to echo at all, so they disable echoing by calling `noecho()`. (See `curs_getch(3CURSES)` for a discussion of how these routines interact with `cbreak()` and `nocbreak()`.)

The `halfdelay()` routine is used for half-delay mode, which is similar to `cbreak()` mode in that characters typed by the user are immediately available to the program. However, after blocking for *tenths* tenths of seconds, `ERR` is returned if nothing has been typed. The value of *tenths* must be a number between 1 and 255. Use `nocbreak()` to leave half-delay mode.

If the `intrflush()` option is enabled, (*bf* is `TRUE`), when an interrupt key is pressed on the keyboard (interrupt, break, quit) all output in the tty driver queue will be flushed, giving the effect of faster response to the interrupt, but causing `curses` to have the wrong idea of what is on the screen. Disabling (*bf* is `FALSE`), the option prevents the flush. The default for the option is inherited from the tty driver settings. The window argument is ignored.

The `keypad()` option enables the keypad of the user's terminal. If enabled (*bf* is `TRUE`), the user can press a function key (such as an arrow key) and `wgetch()` returns a single value representing the function key, as in `KEY_LEFT`. If disabled (*bf* is `FALSE`), `curses` does not treat function keys specially and the program has to interpret the escape sequences itself. If the keypad in the terminal can be turned on (made to transmit) and off (made to work locally), turning on this option causes the terminal keypad to be turned on when `wgetch()` is called. The default value for keypad is false.

Initially, whether the terminal returns 7 or 8 significant bits on input depends on the control mode of the tty driver (see `termio(7I)`). To force 8 bits to be returned, invoke `meta(win, TRUE)`. To force 7 bits to be returned, invoke `meta(win, FALSE)`. The window argument, *win*, is always ignored. If the terminfo capabilities `smm` (`meta_on`) and `rmm` (`meta_off`) are defined for the terminal, `smm` is sent to the terminal when `meta(win, TRUE)` is called and `rmm` is sent when `meta(win, FALSE)` is called.

The `nodelay()` option causes `getch()` to be a non-blocking call. If no input is ready, `getch()` returns `ERR`. If disabled (*bf* is `FALSE`), `getch()` waits until a key is pressed.

While interpreting an input escape sequence, `wgetch()` sets a timer while waiting for the next character. If `notimeout(win, TRUE)` is called, then `wgetch()` does not set a timer. The purpose of the timeout is to differentiate between sequences received from a function key and those typed by a user.

With the `raw()` and `noraw()` routines, the terminal is placed into or out of raw mode. Raw mode is similar to `cbreak()` mode, in that characters typed are immediately passed through to the user program. The differences are that in raw mode, the interrupt, quit, suspend, and flow control characters are all passed through uninterpreted, instead of generating a signal. The behavior of the `BREAK` key depends on other bits in the tty driver that are not set by `curses`.

noqiflush(3CURSES)

When the `noqiflush()` routine is used, normal flush of input and output queues associated with the `INTR`, `QUIT` and `SUSP` characters will not be done (see `termio(7I)`). When `qiflush()` is called, the queues will be flushed when these control characters are read.

The `timeout()` and `wtimeout()` routines set blocking or non-blocking read for a given window. If *delay* is negative, blocking read is used (that is, waits indefinitely for input). If *delay* is zero, then non-blocking read is used (that is, read returns `ERR` if no input is waiting). If *delay* is positive, then read blocks for *delay* milliseconds, and returns `ERR` if there is still no input. Hence, these routines provide the same functionality as `nodelay()`, plus the additional capability of being able to block for only *delay* milliseconds (where *delay* is positive).

`curses` does "line-breakout optimization" by looking for typeahead periodically while updating the screen. If input is found, and it is coming from a tty, the current update is postponed until `refresh()` or `doupdate()` is called again. This allows faster response to commands typed in advance. Normally, the input FILE pointer passed to `newterm()`, or `stdin` in the case that `initscr()` was used, will be used to do this typeahead checking. The `typeahead()` routine specifies that the file descriptor *fildev* is to be used to check for typeahead instead. If *fildev* is `-1`, then no typeahead checking is done.

RETURN VALUES

All routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_getch(3CURSES)`, `curs_initscr(3CURSES)`, `curses(3CURSES)`, `attributes(5)`, `termio(7I)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `echo()`, `noecho()`, `halfdelay()`, `intrflush()`, `meta()`, `nodelay()`, `notimeout()`, `noqiflush()`, `qiflush()`, `timeout()`, and `wtimeout()` may be macros.

| | |
|----------------------|--|
| NAME | noqiflush, qiflush – control flush of input and output on interrupt |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> void noqiflush(void); void qiflush(void);</pre> |
| DESCRIPTION | The qiflush() function enables the flushing of input and output queues when an interrupt, quit, or suspend character is sent to the terminal. The noqiflush() function disables this flushing. |
| RETURN VALUES | These functions do not return a value. |
| ERRORS | None |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: |

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO flushinp(3XCURSES), intrflush(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

noraw(3CURSES)

| | |
|--------------------|---|
| NAME | <code>curs_inopts</code> , <code>cbreak</code> , <code>nocbreak</code> , <code>echo</code> , <code>noecho</code> , <code>halfdelay</code> , <code>intrflush</code> , <code>keypad</code> , <code>meta</code> , <code>nodelay</code> , <code>notimeout</code> , <code>raw</code> , <code>noraw</code> , <code>noqiflush</code> , <code>qiflush</code> , <code>timeout</code> , <code>wtimeout</code> , <code>typeahead</code> – curses terminal input option control routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int cbreak(void); int nocbreak(void); int echo(void); int noecho(void); int halfdelay(int <i>tenths</i>); int intrflush(WINDOW *<i>win</i>, bool <i>bf</i>); int keypad(WINDOW *<i>win</i>, bool <i>bf</i>); int meta(WINDOW *<i>win</i>, bool <i>bf</i>); int nodelay(WINDOW *<i>win</i>, bool <i>bf</i>); int notimeout(WINDOW *<i>win</i>, bool <i>bf</i>); int raw(void); int noraw(void); void noqiflush(void); void qiflush(void); void timeout(int <i>delay</i>); void wtimeout(WINDOW *<i>win</i>, int <i>delay</i>); int typeahead(int <i>fildest</i>);</pre> |
| DESCRIPTION | <p>The <code>cbreak()</code> and <code>nocbreak()</code> routines put the terminal into and out of <code>cbreak()</code> mode, respectively. In this mode, characters typed by the user are immediately available to the program, and erase/kill character-processing is not performed. When out of this mode, the tty driver buffers the typed characters until a newline or carriage return is typed. Interrupt and flow control characters are unaffected by this mode. Initially the terminal may or may not be in <code>cbreak()</code> mode, as the mode is inherited; therefore, a program should call <code>cbreak()</code> or <code>nocbreak()</code> explicitly. Most interactive programs using curses set the <code>cbreak()</code> mode.</p> <p>Note that <code>cbreak()</code> overrides <code>raw()</code>. (See <code>curs_getch(3CURSES)</code> for a discussion of how these routines interact with <code>echo()</code> and <code>noecho()</code>.)</p> <p>The <code>echo()</code> and <code>noecho()</code> routines control whether characters typed by the user are echoed by <code>getch()</code> as they are typed. Echoing by the tty driver is always disabled, but initially <code>getch()</code> is in echo mode, so characters typed are echoed. Authors of</p> |

most interactive programs prefer to do their own echoing in a controlled area of the screen, or not to echo at all, so they disable echoing by calling `noecho()`. (See `curs_getch(3CURSES)` for a discussion of how these routines interact with `cbreak()` and `nocbreak()`.)

The `halfdelay()` routine is used for half-delay mode, which is similar to `cbreak()` mode in that characters typed by the user are immediately available to the program. However, after blocking for *tenths* tenths of seconds, `ERR` is returned if nothing has been typed. The value of *tenths* must be a number between 1 and 255. Use `nocbreak()` to leave half-delay mode.

If the `intrflush()` option is enabled, (*bf* is `TRUE`), when an interrupt key is pressed on the keyboard (interrupt, break, quit) all output in the tty driver queue will be flushed, giving the effect of faster response to the interrupt, but causing curses to have the wrong idea of what is on the screen. Disabling (*bf* is `FALSE`), the option prevents the flush. The default for the option is inherited from the tty driver settings. The window argument is ignored.

The `keypad()` option enables the keypad of the user's terminal. If enabled (*bf* is `TRUE`), the user can press a function key (such as an arrow key) and `wgetch()` returns a single value representing the function key, as in `KEY_LEFT`. If disabled (*bf* is `FALSE`), curses does not treat function keys specially and the program has to interpret the escape sequences itself. If the keypad in the terminal can be turned on (made to transmit) and off (made to work locally), turning on this option causes the terminal keypad to be turned on when `wgetch()` is called. The default value for keypad is false.

Initially, whether the terminal returns 7 or 8 significant bits on input depends on the control mode of the tty driver (see `termio(7I)`). To force 8 bits to be returned, invoke `meta(win, TRUE)`. To force 7 bits to be returned, invoke `meta(win, FALSE)`. The window argument, *win*, is always ignored. If the terminfo capabilities `smm` (`meta_on`) and `rmm` (`meta_off`) are defined for the terminal, `smm` is sent to the terminal when `meta(win, TRUE)` is called and `rmm` is sent when `meta(win, FALSE)` is called.

The `nodelay()` option causes `getch()` to be a non-blocking call. If no input is ready, `getch()` returns `ERR`. If disabled (*bf* is `FALSE`), `getch()` waits until a key is pressed.

While interpreting an input escape sequence, `wgetch()` sets a timer while waiting for the next character. If `notimeout(win, TRUE)` is called, then `wgetch()` does not set a timer. The purpose of the timeout is to differentiate between sequences received from a function key and those typed by a user.

With the `raw()` and `noraw()` routines, the terminal is placed into or out of raw mode. Raw mode is similar to `cbreak()` mode, in that characters typed are immediately passed through to the user program. The differences are that in raw mode, the interrupt, quit, suspend, and flow control characters are all passed through uninterpreted, instead of generating a signal. The behavior of the BREAK key depends on other bits in the tty driver that are not set by `curses`.

noraw(3CURSES)

When the `noqiflush()` routine is used, normal flush of input and output queues associated with the `INTR`, `QUIT` and `SUSP` characters will not be done (see `termio(7I)`). When `qiflush()` is called, the queues will be flushed when these control characters are read.

The `timeout()` and `wtimeout()` routines set blocking or non-blocking read for a given window. If *delay* is negative, blocking read is used (that is, waits indefinitely for input). If *delay* is zero, then non-blocking read is used (that is, read returns `ERR` if no input is waiting). If *delay* is positive, then read blocks for *delay* milliseconds, and returns `ERR` if there is still no input. Hence, these routines provide the same functionality as `nodelay()`, plus the additional capability of being able to block for only *delay* milliseconds (where *delay* is positive).

`curses` does "line-breakout optimization" by looking for typeahead periodically while updating the screen. If input is found, and it is coming from a `tty`, the current update is postponed until `refresh()` or `doupdate()` is called again. This allows faster response to commands typed in advance. Normally, the input `FILE` pointer passed to `newterm()`, or `stdin` in the case that `initscr()` was used, will be used to do this typeahead checking. The `typeahead()` routine specifies that the file descriptor *fildev* is to be used to check for typeahead instead. If *fildev* is `-1`, then no typeahead checking is done.

RETURN VALUES

All routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_getch(3CURSES)`, `curs_initscr(3CURSES)`, `curses(3CURSES)`, `attributes(5)`, `termio(7I)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `echo()`, `noecho()`, `halfdelay()`, `intrflush()`, `meta()`, `nodelay()`, `notimeout()`, `noqiflush()`, `qiflush()`, `timeout()`, and `wtimeout()` may be macros.

| NAME | cbreak, nocbreak, noraw, raw – set input mode controls | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int cbreak(void); int nocbreak(void); int noraw(void); int raw(void);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>cbreak()</code> function enables the character input mode. This overrides any previous call to the <code>raw()</code> function and turns the <code>stty</code> flag <code>ICANON</code> off.</p> <p>The <code>nocbreak()</code> function sets the line canonical mode and turns the <code>stty</code> flag <code>ICANON</code> on without touching the <code>ISIG</code> or <code>IXON</code> flags.</p> <p>The <code>noraw()</code> function sets the line canonical mode and turns the the <code>stty</code> flags <code>ICANON</code>, <code>ISIG</code>, and <code>IXON</code> all on.</p> <p>The <code>raw()</code> function sets the character input mode and turns the <code>stty</code> flags <code>ICANON</code>, <code>ISIG</code>, and <code>IXON</code> all off. This mode provides maximum control over input.</p> <p>It is important to remember that the terminal may or may not be in character mode operation initially. Most interactive programs require <code>cbreak()</code> to be enabled.</p> | | | | | | |
| RETURN VALUES | On success, these functions return <code>OK</code> . Otherwise, they return <code>ERR</code> . | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>getch(3XCURSES)</code> , <code>halfdelay(3XCURSES)</code> , <code>nodelay(3XCURSES)</code> , <code>timeout(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> , <code>termio(7I)</code> | | | | | | |

notimeout(3CURSES)

| | |
|--------------------|---|
| NAME | <code>curs_inopts</code> , <code>cbreak</code> , <code>nocbreak</code> , <code>echo</code> , <code>noecho</code> , <code>halfdelay</code> , <code>intrflush</code> , <code>keypad</code> , <code>meta</code> , <code>nodelay</code> , <code>notimeout</code> , <code>raw</code> , <code>noraw</code> , <code>noqiflush</code> , <code>qiflush</code> , <code>timeout</code> , <code>wtimeout</code> , <code>typeahead</code> – curses terminal input option control routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int cbreak(void); int nocbreak(void); int echo(void); int noecho(void); int halfdelay(int <i>tenths</i>); int intrflush(WINDOW *<i>win</i>, bool <i>bf</i>); int keypad(WINDOW *<i>win</i>, bool <i>bf</i>); int meta(WINDOW *<i>win</i>, bool <i>bf</i>); int nodelay(WINDOW *<i>win</i>, bool <i>bf</i>); int notimeout(WINDOW *<i>win</i>, bool <i>bf</i>); int raw(void); int noraw(void); void noqiflush(void); void qiflush(void); void timeout(int <i>delay</i>); void wtimeout(WINDOW *<i>win</i>, int <i>delay</i>); int typeahead(int <i>fildes</i>);</pre> |
| DESCRIPTION | <p>The <code>cbreak()</code> and <code>nocbreak()</code> routines put the terminal into and out of <code>cbreak()</code> mode, respectively. In this mode, characters typed by the user are immediately available to the program, and erase/kill character-processing is not performed. When out of this mode, the tty driver buffers the typed characters until a newline or carriage return is typed. Interrupt and flow control characters are unaffected by this mode. Initially the terminal may or may not be in <code>cbreak()</code> mode, as the mode is inherited; therefore, a program should call <code>cbreak()</code> or <code>nocbreak()</code> explicitly. Most interactive programs using curses set the <code>cbreak()</code> mode.</p> <p>Note that <code>cbreak()</code> overrides <code>raw()</code>. (See <code>curs_getch(3CURSES)</code> for a discussion of how these routines interact with <code>echo()</code> and <code>noecho()</code>.)</p> <p>The <code>echo()</code> and <code>noecho()</code> routines control whether characters typed by the user are echoed by <code>getch()</code> as they are typed. Echoing by the tty driver is always disabled, but initially <code>getch()</code> is in echo mode, so characters typed are echoed. Authors of</p> |

most interactive programs prefer to do their own echoing in a controlled area of the screen, or not to echo at all, so they disable echoing by calling `noecho()`. (See `curs_getch(3CURSES)` for a discussion of how these routines interact with `cbreak()` and `nocbreak()`.)

The `halfdelay()` routine is used for half-delay mode, which is similar to `cbreak()` mode in that characters typed by the user are immediately available to the program. However, after blocking for *tenths* tenths of seconds, `ERR` is returned if nothing has been typed. The value of *tenths* must be a number between 1 and 255. Use `nocbreak()` to leave half-delay mode.

If the `intrflush()` option is enabled, (*bf* is `TRUE`), when an interrupt key is pressed on the keyboard (interrupt, break, quit) all output in the tty driver queue will be flushed, giving the effect of faster response to the interrupt, but causing `curses` to have the wrong idea of what is on the screen. Disabling (*bf* is `FALSE`), the option prevents the flush. The default for the option is inherited from the tty driver settings. The window argument is ignored.

The `keypad()` option enables the keypad of the user's terminal. If enabled (*bf* is `TRUE`), the user can press a function key (such as an arrow key) and `wgetch()` returns a single value representing the function key, as in `KEY_LEFT`. If disabled (*bf* is `FALSE`), `curses` does not treat function keys specially and the program has to interpret the escape sequences itself. If the keypad in the terminal can be turned on (made to transmit) and off (made to work locally), turning on this option causes the terminal keypad to be turned on when `wgetch()` is called. The default value for keypad is false.

Initially, whether the terminal returns 7 or 8 significant bits on input depends on the control mode of the tty driver (see `termio(7I)`). To force 8 bits to be returned, invoke `meta(win, TRUE)`. To force 7 bits to be returned, invoke `meta(win, FALSE)`. The window argument, *win*, is always ignored. If the terminfo capabilities `smm` (`meta_on`) and `rmm` (`meta_off`) are defined for the terminal, `smm` is sent to the terminal when `meta(win, TRUE)` is called and `rmm` is sent when `meta(win, FALSE)` is called.

The `nodelay()` option causes `getch()` to be a non-blocking call. If no input is ready, `getch()` returns `ERR`. If disabled (*bf* is `FALSE`), `getch()` waits until a key is pressed.

While interpreting an input escape sequence, `wgetch()` sets a timer while waiting for the next character. If `notimeout(win, TRUE)` is called, then `wgetch()` does not set a timer. The purpose of the timeout is to differentiate between sequences received from a function key and those typed by a user.

With the `raw()` and `noraw()` routines, the terminal is placed into or out of raw mode. Raw mode is similar to `cbreak()` mode, in that characters typed are immediately passed through to the user program. The differences are that in raw mode, the interrupt, quit, suspend, and flow control characters are all passed through uninterpreted, instead of generating a signal. The behavior of the `BREAK` key depends on other bits in the tty driver that are not set by `curses`.

notimeout(3CURSES)

When the `noqiflush()` routine is used, normal flush of input and output queues associated with the `INTR`, `QUIT` and `SUSP` characters will not be done (see `termio(7I)`). When `qiflush()` is called, the queues will be flushed when these control characters are read.

The `timeout()` and `wtimeout()` routines set blocking or non-blocking read for a given window. If *delay* is negative, blocking read is used (that is, waits indefinitely for input). If *delay* is zero, then non-blocking read is used (that is, read returns `ERR` if no input is waiting). If *delay* is positive, then read blocks for *delay* milliseconds, and returns `ERR` if there is still no input. Hence, these routines provide the same functionality as `nodelay()`, plus the additional capability of being able to block for only *delay* milliseconds (where *delay* is positive).

`curses` does "line-breakout optimization" by looking for typeahead periodically while updating the screen. If input is found, and it is coming from a tty, the current update is postponed until `refresh()` or `doupdate()` is called again. This allows faster response to commands typed in advance. Normally, the input FILE pointer passed to `newterm()`, or `stdin` in the case that `initscr()` was used, will be used to do this typeahead checking. The `typeahead()` routine specifies that the file descriptor *fildev* is to be used to check for typeahead instead. If *fildev* is `-1`, then no typeahead checking is done.

RETURN VALUES

All routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_getch(3CURSES)`, `curs_initscr(3CURSES)`, `curses(3CURSES)`, `attributes(5)`, `termio(7I)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `echo()`, `noecho()`, `halfdelay()`, `intrflush()`, `meta()`, `nodelay()`, `notimeout()`, `noqiflush()`, `qiflush()`, `timeout()`, and `wtimeout()` may be macros.

NAME | notimeout, timeout, wtimeout – set timed blocking or non-blocking read

SYNOPSIS | `cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \`
`-R /usr/xpg4/lib -lcurses [library...]`

PARAMETERS | `c89 [flag...] file... -lcurses [library...]`

DESCRIPTION | `#include <curses.h>`
`int notimeout(WINDOW *win, bool bf);`
`void timeout(int delay);`
`void wtimeout(WINDOW win, int delay);`

PARAMETERS | *win* | Is a pointer to the window in which to set the timed blocking.
bf | Is a Boolean expression.
delay | Is the number of milliseconds to block or wait for input.

DESCRIPTION | If *bool* is TRUE, the notimeout() function disables a timer used by getch(3XCURSES) when handling multibyte function key sequences.

When *bool* is FALSE and keypad handling is enabled, a timer is set by getch() to handle bytes received that could be the beginning of a function key (for example, ESC). If the remainder of the sequence is not received before the time expires, the first byte is returned; otherwise, the value of the function key is returned. Subsequent calls to the getch() function will return the other bytes received for the incomplete key sequence.

The timeout() and wtimeout() functions set the length of time getch() waits for input for windows stdscr and win, respectively. These functions are similar to nodelay(3XCURSES) except the time to block or wait for input can be specified.

A negative *delay* causes the program to wait indefinitely for input; a *delay* of 0 returns ERR if no input is ready; and a positive *delay* blocks until input arrives or the time specified expires, (in which case, ERR is returned).

RETURN VALUES | On success, the notimeout() function returns OK. Otherwise, it returns ERR.

The timeout() and wtimeout() functions do not return a value.

ERRORS | None.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |

notimeout(3XCURSES)

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `getch(3XCURSES)`, `halfdelay(3XCURSES)`, `libcurses(3XCURSES)`,
`nodelay(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | plot, arc, box, circle, closepl, closevt, cont, erase, label, line, linemod, move, openpl, openvt, point, space – graphics interface |
| SYNOPSIS | <pre>cc [flag ...] file ... -lplot [library...] #include <plot.h> void arc(short x0, short y0, short x1, short y1, short x2, short y2); void box(short x0, short y0, short x1, short y1); void circle(short x, short y, short r); void closepl(); void closevt(); void cont(short x, short y); void erase(); void label(char *s); void line(short x0, short y0, short x1, short y1); void linemod(char *s); void move(short x, short y); void openpl(); void openvt(); void point(short x, short y); void space(short x0, short y0, short x1, short y1);</pre> |
| DESCRIPTION | <p>These functions generate graphics output for a set of output devices. The format of the output is dependent upon which link editor option is used when the program is compiled and linked (see Link Editor).</p> <p>The term "current point" refers to the current setting for the x and y coordinates.</p> <p>The <code>arc()</code> function specifies a circular arc. The coordinates $(x0, y0)$ specify the center of the arc. The coordinates $(x1, y1)$ specify the starting point of the arc. The coordinates $(x2, y2)$ specify the end point of the circular arc.</p> <p>The <code>box()</code> function specifies a rectangle with coordinates $(x0, y0)$, $(x0, y1)$, $(x1, y0)$, and $(x1, y1)$. The current point is set to $(x1, y1)$.</p> <p>The <code>circle()</code> function specifies a circle with a center at the coordinates (x, y) and a radius of r.</p> <p>The <code>closevt()</code> and <code>closepl()</code> functions flush the output.</p> <p>The <code>cont()</code> function specifies a line beginning at the current point and ending at the coordinates (x, y). The current point is set to (x, y).</p> |

openpl(3PLOT)

The `erase()` function starts another frame of output.

The `label()` function places the null terminated string `s` so that the first character falls on the current point. The string is then terminated by a NEWLINE character.

The `line()` function draws a line starting at the coordinates (x_0, y_0) and ending at the coordinates (x_1, y_1) . The current point is set to (x_1, y_1) .

The `linemod()` function specifies the style for drawing future lines. `s` may contain one of the following: `dotted`, `solid`, `longdashed`, `shortdashed`, or `dotdashed`.

The `move()` function sets the current point to the coordinates (x, y) .

The `openpl()` or `openvt()` function must be called to open the device before any other plot functions are called.

The `point()` function plots the point given by the coordinates (x, y) . The current point is set to (x, y) .

The `space()` function specifies the size of the plotting area. The plot will be reduced or enlarged as necessary to fit the area specified. The coordinates (x_0, y_0) specify the lower left hand corner of the plotting area. The coordinates (x_1, y_1) specify the upper right hand corner of the plotting area.

Link Editor

Various flavors of these functions exist for different output devices. They are obtained by using the following `ld(1)` options:

| | |
|---------------------|--|
| <code>-lplot</code> | device-independent graphics stream on standard output in the format described in <code>plot(4B)</code> |
| <code>-l300</code> | GSI 300 terminal |
| <code>-l300s</code> | GSI 300S terminal |
| <code>-l4014</code> | Tektronix 4014 terminal |
| <code>-l450</code> | GSI 450 terminal |
| <code>-lvt0</code> | |

FILES

| | |
|--|----------------------|
| <code>/usr/lib/libplot.a</code> | archive library |
| <code>/usr/lib/libplot.so.1</code> | shared object |
| <code>/usr/lib/sparcv9/libplot.so.1</code> | 64-bit shared object |
| <code>/usr/lib/lib300.a</code> | archive library |
| <code>/usr/lib/lib300.so.1</code> | shared object |

```

/usr/lib/sparcv9/lib300.so.1
  64-bit shared object
/usr/lib/lib300s.a
  archive library
/usr/lib/lib300s.so.1
  shared object
/usr/lib/sparcv9/lib300s.so.1
  64-bit shared object
/usr/lib/lib4014.a
  archive library
/usr/lib/lib4014.so.1
  shared object
/usr/lib/sparcv9/lib4014.so.1
  64-bit shared object
/usr/lib/lib450.a
  archive library
/usr/lib/lib450.so.1
  shared object
/usr/lib/sparcv9/lib450.so.1
  64-bit shared object
/usr/lib/libvt0.a
  archive library
/usr/lib/libvt0.so.1
  shared object
/usr/lib/sparcv9/libvt0.so.1
  64-bit shared object

```

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `graph(1)`, `ld(1)`, `libplot(3LIB)`, `plot(4B)`, `attributes(5)`

openvt(3PLOT)

| | |
|--------------------|--|
| NAME | plot, arc, box, circle, closepl, closevt, cont, erase, label, line, linemod, move, openpl, openvt, point, space – graphics interface |
| SYNOPSIS | <pre>cc [flag ...] file ... -lplot [library...] #include <plot.h> void arc(short x0, short y0, short x1, short y1, short x2, short y2); void box(short x0, short y0, short x1, short y1); void circle(short x, short y, short r); void closepl(); void closevt(); void cont(short x, short y); void erase(); void label(char *s); void line(short x0, short y0, short x1, short y1); void linemod(char *s); void move(short x, short y); void openpl(); void openvt(); void point(short x, short y); void space(short x0, short y0, short x1, short y1);</pre> |
| DESCRIPTION | <p>These functions generate graphics output for a set of output devices. The format of the output is dependent upon which link editor option is used when the program is compiled and linked (see Link Editor).</p> <p>The term "current point" refers to the current setting for the x and y coordinates.</p> <p>The <code>arc()</code> function specifies a circular arc. The coordinates $(x0, y0)$ specify the center of the arc. The coordinates $(x1, y1)$ specify the starting point of the arc. The coordinates $(x2, y2)$ specify the end point of the circular arc.</p> <p>The <code>box()</code> function specifies a rectangle with coordinates $(x0, y0)$, $(x0, y1)$, $(x1, y0)$, and $(x1, y1)$. The current point is set to $(x1, y1)$.</p> <p>The <code>circle()</code> function specifies a circle with a center at the coordinates (x, y) and a radius of r.</p> <p>The <code>closevt()</code> and <code>closepl()</code> functions flush the output.</p> <p>The <code>cont()</code> function specifies a line beginning at the current point and ending at the coordinates (x, y). The current point is set to (x, y).</p> |

The `erase()` function starts another frame of output.

The `label()` function places the null terminated string `s` so that the first character falls on the current point. The string is then terminated by a NEWLINE character.

The `line()` function draws a line starting at the coordinates $(x0, y0)$ and ending at the coordinates $(x1, y1)$. The current point is set to $(x1, y1)$.

The `linemod()` function specifies the style for drawing future lines. `s` may contain one of the following: `dotted`, `solid`, `longdashed`, `shortdashed`, or `dotdashed`.

The `move()` function sets the current point to the coordinates (x, y) .

The `openpl()` or `openvt()` function must be called to open the device before any other plot functions are called.

The `point()` function plots the point given by the coordinates (x, y) . The current point is set to (x, y) .

The `space()` function specifies the size of the plotting area. The plot will be reduced or enlarged as necessary to fit the area specified. The coordinates $(x0, y0)$ specify the lower left hand corner of the plotting area. The coordinates $(x1, y1)$ specify the upper right hand corner of the plotting area.

Link Editor Various flavors of these functions exist for different output devices. They are obtained by using the following `ld(1)` options:

```
-lplot          device-independent graphics stream on standard output in the
                 format described in plot(4B)
-l300           GSI 300 terminal
-l300s          GSI 300S terminal
-l4014          Tektronix 4014 terminal
-l450           GSI 450 terminal
-lvt0
```

FILES

```
/usr/lib/libplot.a
  archive library
/usr/lib/libplot.so.1
  shared object
/usr/lib/sparcv9/libplot.so.1
  64-bit shared object
/usr/lib/lib300.a
  archive library
/usr/lib/lib300.so.1
  shared object
```

openvt(3PLOT)

/usr/lib/sparcv9/lib300.so.1
64-bit shared object

/usr/lib/lib300s.a
archive library

/usr/lib/lib300s.so.1
shared object

/usr/lib/sparcv9/lib300s.so.1
64-bit shared object

/usr/lib/lib4014.a
archive library

/usr/lib/lib4014.so.1
shared object

/usr/lib/sparcv9/lib4014.so.1
64-bit shared object

/usr/lib/lib450.a
archive library

/usr/lib/lib450.so.1
shared object

/usr/lib/sparcv9/lib450.so.1
64-bit shared object

/usr/lib/libvt0.a
archive library

/usr/lib/libvt0.so.1
shared object

/usr/lib/sparcv9/libvt0.so.1
64-bit shared object

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `graph(1)`, `ld(1)`, `libplot(3LIB)`, `plot(4B)`, `attributes(5)`

NAME curs_overlay, overlay, overwrite, copywin – overlap and manipulate overlapped curses windows

SYNOPSIS

```
cc [ flag ... ] file ... -lcurses [ library ... ]
#include <curses.h>

int overlay(WINDOW *srcwin, WINDOW *dstwin);
int overwrite(WINDOW *srcwin, WINDOW *dstwin);
int copywin(WINDOW *srcwin, WINDOW *dstwin, int sminrow, int smincol,
            int dminrow, int dmincol, int dmaxrow, int dmaxcol, int overlay);
```

DESCRIPTION The overlay() and overwrite() routines overlay *srcwin* on top of *dstwin*. *srcwin* and *dstwin* are not required to be the same size; only text where the two windows overlap is copied. The difference is that overlay() is non-destructive (blanks are not copied) whereas overwrite() is destructive.

The copywin() routine provides a finer granularity of control over the overlay() and overwrite() routines. Like in the prefresh() routine, a rectangle is specified in the destination window, (*dminrow*, *dmincol*) and (*dmaxrow*, *dmaxcol*), and the upper-left-corner coordinates of the source window, (*sminrow*, *smincol*). If the argument *overlay* is true, then copying is non-destructive, as in overlay().

RETURN VALUES Routines that return an integer return ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_pad(3CURSES), curs_refresh(3CURSES), curses(3CURSES), attributes(5)

NOTES The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.

Note that overlay() and overwrite may be macros.

overlay(3XCURSES)

| | |
|----------------------|---|
| NAME | overlay, overwrite – copy overlapped windows |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int overlay(const WINDOW *srcwin, WINDOW *dstwin); int overwrite(const WINDOW *srcwin, WINDOW *dstwin);</pre> |
| PARAMETERS | <p><i>srcwin</i> Is a pointer to the source window to be copied.</p> <p><i>dstwin</i> Is a pointer to the destination window to be overlaid or overwritten.</p> |
| DESCRIPTION | <p>The <code>overwrite()</code> and <code>overlay()</code> functions overlay <i>srcwin</i> on top of <i>dstwin</i>. The <i>srcwin</i> and <i>dstwin</i> arguments do not have to be the same size; only text where the two windows overlap is copied.</p> <p>The <code>overwrite()</code> function copies characters as though a sequence of <code>win_wch(3XCURSES)</code> and <code>wadd_wch(3XCURSES)</code> were performed with the destination window's attributes and background attributes cleared.</p> <p>The <code>overlay()</code> function does the same thing, except that, whenever a character to be copied is the background character of the source window, <code>overlay()</code> does not copy the character but merely moves the destination cursor the width of the source background character.</p> <p>If any portion of the overlaying window border is not the first column of a multi-column character, then all the column positions will be replaced with the background character and rendition before the overlay is done. If the default background character is a multi-column character when this occurs, then these functions fail.</p> |
| RETURN VALUES | Upon successful completion, these functions return OK. Otherwise, they return ERR. |
| ERRORS | No errors are defined. |
| EXAMPLES | <p>EXAMPLE 1 Implement a pop-up dialog</p> <p>The following example demonstrates the use of <code>overwrite()</code> to implement a pop-up dialog box.</p> <pre>#include <curses.h> /* * Pop-up a window on top of curscr. If row and/or col * are -1 then that dimension will be centered within * curscr. Return 0 for success or -1 if malloc() failed. * Pass back the working window and the saved window for the * pop-up. The saved window should not be modified. */</pre> |

EXAMPLE 1 Implement a pop-up dialog (Continued)

```

int
popup(work, save, nrows, ncols, row, col)
WINDOW **work, **save;
int nrows, ncols, row, col;
{
    int mr, mc;
    getmaxyx(curscr, mr, mc);
    /* Windows are limited to the size of curscr. */
    if (mr < nrows)
        nrows = mr;
    if (mc < ncols)
        ncols = mc;
    /* Center dimensions. */
    if (row == -1)
        row = (mr-nrows)/2;
    if (col == -1)
        col = (mc-ncols)/2;
    /* The window must fit entirely in curscr. */
    if (mr < row+nrows)
        row = 0;
    if (mc < col+ncols)
        col = 0;
    *work = newwin(nrows, ncols, row, col);
    if (*work == NULL)
        return (-1);
    if ((*save = dupwin(*work)) == NULL) {
        delwin(*work);
        return (-1);
    }
    overwrite(curscr, *save);
    return (0);
}
/*
 * Restore the region covered by a pop-up window.
 * Delete the working window and the saved window.
 * This function is the complement to popup(). Return
 * 0 for success or -1 for an error.
 */
int
popdown(work, save)
WINDOW *work, *save;
{
    (void) wnoutrefresh(save);
    (void) delwin(save);
    (void) delwin(work);
    return (0);
}
/*
 * Compute the size of a dialog box that would fit around
 * the string.
 */
void
dialsize(str, nrows, ncols)
char *str;

```

EXAMPLE 1 Implement a pop-up dialog *(Continued)*

```

int *nrows, *ncols;
{
    int rows, cols, col;
    for (rows = 1, cols = col = 0; *str != '\0'; ++str) {
        if (*str == '\n') {
            if (cols < col)
                cols = col;
            col = 0;
            ++rows;
        } else {
            ++col;
        }
    }
    if (cols < col)
        cols = col;
    *nrows = rows;
    *ncols = cols;
}
/*
 * Write a string into a dialog box.
 */
void
dialfill(w, s)
WINDOW *w;
char *s;
{
    int row;
    (void) wmove(w, 1, 1);
    for (row = 1; *s != '\0'; ++s) {
        (void) waddch(w, *((unsigned char*) s));
        if (*s == '\n')
            wmove(w, ++row, 1);
    }
    box(w, 0, 0);
}
void
dialog(str)
char *str;
{
    WINDOW *work, *save;
    int nrows, ncols, row, col;
    /* Figure out size of window. */
    dialsize(str, &nrows, &ncols);
    /* Create a centered working window with extra */
    /* room for a border. */
    (void) popup(&work, &save, nrows+2, ncols+2, -1, -1);
    /* Write text into the working window. */
    dialfill(work, str);
    /* Pause. Remember that wgetch( ) will do a wrefresh( ) */
    /* for us. */
    (void) wgetch(work);
    /* Restore curscr and free windows. */
    (void) popdown(work, save);
    /* Redraw curscr to remove window from physical screen. */
}

```

EXAMPLE 1 Implement a pop-up dialog *(Continued)*

```
(void) douupdate( );
}
```

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `copywin(3XCURSES)`, `libcurses(3XCURSES)`, `wadd_wch(3XCURSES)`, `win_wch(3XCURSES)`, `attributes(5)`, `standards(5)`

overwrite(3CURSES)

NAME curs_overlay, overlay, overwrite, copywin – overlap and manipulate overlapped curses windows

SYNOPSIS

```
cc [ flag ... ] file ... -lcurses [ library ... ]
#include <curses.h>

int overlay(WINDOW *srcwin, WINDOW *dstwin);
int overwrite(WINDOW *srcwin, WINDOW *dstwin);

int copywin(WINDOW *srcwin, WINDOW *dstwin, int sminrow, int smincol,
            int dminrow, int dmincol, int dmaxrow, int dmaxcol, int overlay);
```

DESCRIPTION The overlay() and overwrite() routines overlay *srcwin* on top of *dstwin*. *srcwin* and *dstwin* are not required to be the same size; only text where the two windows overlap is copied. The difference is that overlay() is non-destructive (blanks are not copied) whereas overwrite() is destructive.

The copywin() routine provides a finer granularity of control over the overlay() and overwrite() routines. Like in the refresh() routine, a rectangle is specified in the destination window, (*dminrow*, *dmincol*) and (*dmaxrow*, *dmaxcol*), and the upper-left-corner coordinates of the source window, (*sminrow*, *smincol*). If the argument *overlay* is true, then copying is non-destructive, as in overlay().

RETURN VALUES Routines that return an integer return ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_pad(3CURSES), curs_refresh(3CURSES), curses(3CURSES), attributes(5)

NOTES The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.

Note that overlay() and overwrite may be macros.

| | |
|----------------------|---|
| NAME | overlay, overwrite – copy overlapped windows |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int overlay(const WINDOW *srcwin, WINDOW *dstwin); int overwrite(const WINDOW *srcwin, WINDOW *dstwin);</pre> |
| PARAMETERS | <p><i>srcwin</i> Is a pointer to the source window to be copied.</p> <p><i>dstwin</i> Is a pointer to the destination window to be overlaid or overwritten.</p> |
| DESCRIPTION | <p>The <code>overwrite()</code> and <code>overlay()</code> functions overlay <i>srcwin</i> on top of <i>dstwin</i>. The <i>srcwin</i> and <i>dstwin</i> arguments do not have to be the same size; only text where the two windows overlap is copied.</p> <p>The <code>overwrite()</code> function copies characters as though a sequence of <code>win_wch(3XCURSES)</code> and <code>wadd_wch(3XCURSES)</code> were performed with the destination window's attributes and background attributes cleared.</p> <p>The <code>overlay()</code> function does the same thing, except that, whenever a character to be copied is the background character of the source window, <code>overlay()</code> does not copy the character but merely moves the destination cursor the width of the source background character.</p> <p>If any portion of the overlaying window border is not the first column of a multi-column character, then all the column positions will be replaced with the background character and rendition before the overlay is done. If the default background character is a multi-column character when this occurs, then these functions fail.</p> |
| RETURN VALUES | Upon successful completion, these functions return OK. Otherwise, they return ERR. |
| ERRORS | No errors are defined. |
| EXAMPLES | <p>EXAMPLE 1 Implement a pop-up dialog</p> <p>The following example demonstrates the use of <code>overwrite()</code> to implement a pop-up dialog box.</p> <pre>#include <curses.h> /* * Pop-up a window on top of curscr. If row and/or col * are -1 then that dimension will be centered within * curscr. Return 0 for success or -1 if malloc() failed. * Pass back the working window and the saved window for the * pop-up. The saved window should not be modified. */</pre> |

overwrite(3XCURSES)

EXAMPLE 1 Implement a pop-up dialog (Continued)

```
int
popup(work, save, nrows, ncols, row, col)
WINDOW **work, **save;
int nrows, ncols, row, col;
{
    int mr, mc;
    getmaxyx(curscr, mr, mc);
    /* Windows are limited to the size of curscr. */
    if (mr < nrows)
        nrows = mr;
    if (mc < ncols)
        ncols = mc;
    /* Center dimensions. */
    if (row == -1)
        row = (mr-nrows)/2;
    if (col == -1)
        col = (mc-ncols)/2;
    /* The window must fit entirely in curscr. */
    if (mr < row+nrows)
        row = 0;
    if (mc < col+ncols)
        col = 0;
    *work = newwin(nrows, ncols, row, col);
    if (*work == NULL)
        return (-1);
    if ((*save = dupwin(*work)) == NULL) {
        delwin(*work);
        return (-1);
    }
    overwrite(curscr, *save);
    return (0);
}
/*
 * Restore the region covered by a pop-up window.
 * Delete the working window and the saved window.
 * This function is the complement to popup(). Return
 * 0 for success or -1 for an error.
 */
int
popdown(work, save)
WINDOW *work, *save;
{
    (void) wnoutrefresh(save);
    (void) delwin(save);
    (void) delwin(work);
    return (0);
}
/*
 * Compute the size of a dialog box that would fit around
 * the string.
 */
void
dialsize(str, nrows, ncols)
char *str;
```

EXAMPLE 1 Implement a pop-up dialog (Continued)

```

int *nrows, *ncols;
{
    int rows, cols, col;
    for (rows = 1, cols = col = 0; *str != '\0'; ++str) {
        if (*str == '\n') {
            if (cols < col)
                cols = col;
            col = 0;
            ++rows;
        } else {
            ++col;
        }
    }
    if (cols < col)
        cols = col;
    *nrows = rows;
    *ncols = cols;
}
/*
 * Write a string into a dialog box.
 */
void
dialfill(w, s)
WINDOW *w;
char *s;
{
    int row;
    (void) wmove(w, 1, 1);
    for (row = 1; *s != '\0'; ++s) {
        (void) waddch(w, *((unsigned char*) s));
        if (*s == '\n')
            wmove(w, ++row, 1);
    }
    box(w, 0, 0);
}
void
dialog(str)
char *str;
{
    WINDOW *work, *save;
    int nrows, ncols, row, col;
    /* Figure out size of window. */
    dialsize(str, &nrows, &ncols);
    /* Create a centered working window with extra */
    /* room for a border. */
    (void) popup(&work, &save, nrows+2, ncols+2, -1, -1);
    /* Write text into the working window. */
    dialfill(work, str);
    /* Pause. Remember that wgetch( ) will do a wrefresh( ) */
    /* for us. */
    (void) wgetch(work);
    /* Restore curscr and free windows. */
    (void) popdown(work, save);
    /* Redraw curscr to remove window from physical screen. */
}

```

overwrite(3XCURSES)

EXAMPLE 1 Implement a pop-up dialog *(Continued)*

```
(void) douupdate( );  
}
```

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `copywin(3XCURSES)`, `libcurses(3XCURSES)`, `wadd_wch(3XCURSES)`, `win_wch(3XCURSES)`, `attributes(5)`, `standards(5)`

pair_content(3CURSES)

| | |
|-----------------------------|--|
| NAME | <code> curs_color, start_color, init_pair, init_color, has_colors, can_change_color, color_content, pair_content </code> – curses color manipulation routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int start_color(void); int init_pair(short pair, short fg, short bg); int init_color(short color, short red, short green, short blue); bool has_colors(void); bool can_change_color(void); int color_content(short color, short *redp, short *greenp, short *bluep); int pair_content(short pair, short *fgp, short *bgp);</pre> |
| Overview | <p>curses provides routines that manipulate color on color alphanumeric terminals. To use these routines <code>start_color()</code> must be called, usually right after <code>initscr()</code>. See <code>curs_initscr(3CURSES)</code>. Colors are always used in pairs (referred to as color-pairs). A color-pair consists of a foreground color (for characters) and a background color (for the field on which the characters are displayed). A programmer initializes a color-pair with the routine <code>init_pair</code>. After it has been initialized, <code>COLOR_PAIR(n)</code>, a macro defined in <code><curses.h></code>, can be used in the same ways other video attributes can be used. If a terminal is capable of redefining colors, the programmer can use the routine <code>init_color()</code> to change the definition of a color. The routines <code>has_colors()</code> and <code>can_change_color()</code> return <code>TRUE</code> or <code>FALSE</code>, depending on whether the terminal has color capabilities and whether the programmer can change the colors. The routine <code>color_content()</code> allows a programmer to identify the amounts of red, green, and blue components in an initialized color. The routine <code>pair_content()</code> allows a programmer to find out how a given color-pair is currently defined.</p> |
| Routine Descriptions | <p>The <code>start_color()</code> routine requires no arguments. It must be called if the programmer wants to use colors, and before any other color manipulation routine is called. It is good practice to call this routine right after <code>initscr()</code>. <code>start_color()</code> initializes eight basic colors (black, red, green, yellow, blue, magenta, cyan, and white), and two global variables, <code>COLORS</code> and <code>COLOR_PAIRS</code> (respectively defining the maximum number of colors and color-pairs the terminal can support). It also restores the colors on the terminal to the values they had when the terminal was just turned on.</p> <p>The <code>init_pair()</code> routine changes the definition of a color-pair. It takes three arguments: the number of the color-pair to be changed, the foreground color number, and the background color number. The value of the first argument must be between 1 and <code>COLOR_PAIRS-1</code>. The value of the second and third arguments must be between 0 and <code>COLORS</code>. If the color-pair was previously initialized, the screen is refreshed and all occurrences of that color-pair is changed to the new definition.</p> |

pair_content(3CURSES)

The `init_color()` routine changes the definition of a color. It takes four arguments: the number of the color to be changed followed by three RGB values (for the amounts of red, green, and blue components). The value of the first argument must be between 0 and `COLORS`. (See the section `COLORS` for the default color index.) Each of the last three arguments must be a value between 0 and 1000. When `init_color()` is used, all occurrences of that color on the screen immediately change to the new definition.

The `has_colors()` routine requires no arguments. It returns `TRUE` if the terminal can manipulate colors; otherwise, it returns `FALSE`. This routine facilitates writing terminal-independent programs. For example, a programmer can use it to decide whether to use color or some other video attribute.

The `can_change_color()` routine requires no arguments. It returns `TRUE` if the terminal supports colors and can change their definitions; other, it returns `FALSE`. This routine facilitates writing terminal-independent programs.

The `color_content()` routine gives users a way to find the intensity of the red, green, and blue (RGB) components in a color. It requires four arguments: the color number, and three addresses of `shorts` for storing the information about the amounts of red, green, and blue components in the given color. The value of the first argument must be between 0 and `COLORS`. The values that are stored at the addresses pointed to by the last three arguments are between 0 (no component) and 1000 (maximum amount of component).

The `pair_content()` routine allows users to find out what colors a given color-pair consists of. It requires three arguments: the color-pair number, and two addresses of `shorts` for storing the foreground and the background color numbers. The value of the first argument must be between 1 and `COLOR_PAIRS-1`. The values that are stored at the addresses pointed to by the second and third arguments are between 0 and `COLORS`.

Colors In `< curses.h >` the following macros are defined. These are the default colors. `curses` also assumes that `COLOR_BLACK` is the default background color for all terminals.

```
COLOR_BLACK
COLOR_RED
COLOR_GREEN
COLOR_YELLOW
COLOR_BLUE
COLOR_MAGENTA
COLOR_CYAN
COLOR_WHITE
```

RETURN VALUES All routines that return an integer return `ERR` upon failure and `OK` upon successful completion.

pair_content(3CURSES)

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_attr(3CURSES), curs_initscr(3CURSES), curses(3CURSES), attributes(5)

NOTES The header < curses . h > automatically includes the headers < stdio . h > and < unctrl . h >.

pair_content(3XCURSES)

| | |
|------------------------------|---|
| NAME | <code>can_change_color</code> , <code>color_content</code> , <code>COLOR_PAIR</code> , <code>has_colors</code> , <code>init_color</code> , <code>init_pair</code> , <code>pair_content</code> , <code>PAIR_NUMBER</code> , <code>start_color</code> , <code>COLOR_PAIRS</code> , <code>COLORS</code> – manipulate color information |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> bool can_change_color(void); int color_content(short color, short *red, short *green, short *blue); int COLOR_PAIR(int n); bool has_colors(void); int init_color(short color, short red, short green, short blue); int init_pair(short pair, short f, short b); int pair_content(short pair, short *f, short *b); int PAIR_NUMBER(int value); int start_color(void); extern int COLOR_PAIRS; extern int COLORS;</pre> |
| DESCRIPTION | These functions manipulate color on terminals that support color. |
| Querying Capabilities | The <code>has_colors()</code> function indicates whether the terminal is a color terminal. The <code>can_change_color()</code> function indicates whether the terminal is a color terminal on which colors can be redefined. |
| Initialization | <p>The <code>start_color()</code> function must be called to enable use of colors and before any color manipulation function is called. The function initializes eight basic colors (black, red, green, yellow, blue, magenta, cyan, and white) that can be specified by the color macros (such as <code>COLOR_BLACK</code>) defined in <code><curses.h></code>. The initial appearance of these colors is unspecified.</p> <p>The function also initializes two global external variables:</p> <ul style="list-style-type: none">■ <code>COLORS</code> defines the number of colors that the terminal supports. See Color Identification below. If <code>COLORS</code> is 0, the terminal does not support redefinition of colors and <code>can_change_color()</code> will return <code>FALSE</code>.■ <code>COLOR_PAIRS</code> defines the maximum number of color-pairs that the terminal supports. See User-defined Color Pairs below. |

pair_content(3XCURSES)

Color Identification

The `start_color()` function also restores the colors on the terminal to terminal-specific initial values. The initial background color is assumed to be black for all terminals.

The `init_color()` function redefines color number *color*, on terminals that support the redefinition of colors, to have the red, green, and blue intensity components specified by *red*, *green*, and *blue*, respectively. Calling `init_color()` also changes all occurrences of the specified color on the screen to the new definition.

The `color_content()` function identifies the intensity components of color number *color*. It stores the red, green, and blue intensity components of this color in the addresses pointed to by *red*, *green*, and *blue*, respectively.

For both functions, the *color* argument must be in the range from 0 to and including `COLORS-1`. Valid intensity value range from 0 (no intensity component) up to and including 1000 (maximum intensity in that component).

User-defined Color Pairs

Calling `init_pair()` defines or redefines color-pair number *pair* to have foreground color *f* and background color *b*. Calling `init_pair()` changes any characters that were displayed in the color pair's old definition to the new definition and refreshes the screen.

After defining the color pair, the macro `COLOR_PAIR(n)` returns the value of color pair *n*. This value is the color attribute as it would be extracted from a `chtype`. Controversy, the macro `COLOR_NUMBER(value)` returns the color pair number associated with the color attribute *value*.

The `pair_content()` retrieves the component colors of a color-pair number *pair*. It stores the foreground and background color numbers in the variables pointed to by *f* and *b*, respectively.

With `init_pair()` and `pair_content()`, the value of *pair* must be in a range from 0 to and including `COLOR_PAIRS-1`. Valid values for *f* and *b* are the range from 0 to and including `COLORS-1`.

PARAMETERS

| | |
|--------------|--|
| <i>color</i> | Is the number of the color for which to provide information (0 to <code>COLORS-1</code>). |
| <i>red</i> | Is a pointer to the RGB value for the amount of red in <i>color</i> . |
| <i>green</i> | Is a pointer to the RGB value for the amount of green in <i>color</i> . |
| <i>blue</i> | Is a pointer to the RGB value for the amount of blue in <i>color</i> . |
| <i>n</i> | Is the number of a color pair. |
| <i>pair</i> | Is the number of the color pair for which to provide information (1 to <code>COLOR_PAIRS-1</code>). |
| <i>f</i> | Is a pointer to the number of the foreground color (0 to <code>COLORS-1</code>) in <i>pair</i> . |

pair_content(3XCURSES)

b Is a pointer to the number of the background color (0 to COLORS-1) in *pair*.

value Is a color attribute value.

RETURN VALUES The `has_colors()` function returns `TRUE` if the terminal can manipulate colors. Otherwise, it returns `FALSE`.

The `can_change_color()` function returns `TRUE` if the terminal supports colors and is able to change their definitions. Otherwise, it returns `FALSE`.

Upon successful completion, the other functions return `OK`. Otherwise, they return `ERR`.

ERRORS No errors are defined.

USAGE To use these functions, `start_color()` must be called, usually right after `initscr(3XCURSES)`.

The `can_change_color()` and `has_colors()` functions facilitate writing terminal-independent applications. For example, a programmer can use them to decide whether to use color or some other video attribute.

On color terminals, a typical value of `COLORS` is 8 and the macros such as `COLOR_BLACK` return a value within the range from 0 to and including 7. However, applications cannot rely on this to be true.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attroff(3XCURSES)`, `delscreen(3XCURSES)`, `initscr(3XCURSES)`, `lib curses(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|------------------------------|--|
| NAME | can_change_color, color_content, COLOR_PAIR, has_colors, init_color, init_pair, pair_content, PAIR_NUMBER, start_color, COLOR_PAIRS, COLORS – manipulate color information |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> bool can_change_color(void); int color_content(short color, short *red, short *green, short *blue); int COLOR_PAIR(int n); bool has_colors(void); int init_color(short color, short red, short green, short blue); int init_pair(short pair, short f, short b); int pair_content(short pair, short *f, short *b); int PAIR_NUMBER(int value); int start_color(void); extern int COLOR_PAIRS; extern int COLORS;</pre> |
| DESCRIPTION | These functions manipulate color on terminals that support color. |
| Querying Capabilities | The has_colors() function indicates whether the terminal is a color terminal. The can_change_color() function indicates whether the terminal is a color terminal on which colors can be redefined. |
| Initialization | <p>The start_color() function must be called to enable use of colors and before any color manipulation function is called. The function initializes eight basic colors (black, red, green, yellow, blue, magenta, cyan, and white) that can be specified by the color macros (such as COLOR_BLACK) defined in <curses.h>. The initial appearance of these colors is unspecified.</p> <p>The function also initializes two global external variables:</p> <ul style="list-style-type: none"> ■ COLORS defines the number of colors that the terminal supports. See Color Identification below. If COLORS is 0, the terminal does not support redefinition of colors and can_change_color() will return FALSE. ■ COLOR_PAIRS defines the maximum number of color-pairs that the terminal supports. See User-defined Color Pairs below. |

PAIR_NUMBER(3XCURSES)

| | | | | | | | | | | | | | | | |
|---------------------------------|--|--------------|--|------------|---|--------------|---|-------------|--|----------|--------------------------------|-------------|--|----------|---|
| | <p>The <code>start_color()</code> function also restores the colors on the terminal to terminal-specific initial values. The initial background color is assumed to be black for all terminals.</p> | | | | | | | | | | | | | | |
| Color Identification | <p>The <code>init_color()</code> function redefines color number <i>color</i>, on terminals that support the redefinition of colors, to have the red, green, and blue intensity components specified by <i>red</i>, <i>green</i>, and <i>blue</i>, respectively. Calling <code>init_color()</code> also changes all occurrences of the specified color on the screen to the new definition.</p> <p>The <code>color_content()</code> function identifies the intensity components of color number <i>color</i>. It stores the red, green, and blue intensity components of this color in the addresses pointed to by <i>red</i>, <i>green</i>, and <i>blue</i>, respectively.</p> <p>For both functions, the <i>color</i> argument must be in the range from 0 to and including <code>COLORS-1</code>. Valid intensity value range from 0 (no intensity component) up to and including 1000 (maximum intensity in that component).</p> | | | | | | | | | | | | | | |
| User-defined Color Pairs | <p>Calling <code>init_pair()</code> defines or redefines color-pair number <i>pair</i> to have foreground color <i>f</i> and background color <i>b</i>. Calling <code>init_pair()</code> changes any characters that were displayed in the color pair's old definition to the new definition and refreshes the screen.</p> <p>After defining the color pair, the macro <code>COLOR_PAIR(n)</code> returns the value of color pair <i>n</i>. This value is the color attribute as it would be extracted from a <code>chtype</code>. Controversy, the macro <code>COLOR_NUMBER(value)</code> returns the color pair number associated with the color attribute <i>value</i>.</p> <p>The <code>pair_content()</code> retrieves the component colors of a color-pair number <i>pair</i>. It stores the foreground and background color numbers in the variables pointed to by <i>f</i> and <i>b</i>, respectively.</p> <p>With <code>init_pair()</code> and <code>pair_content()</code>, the value of <i>pair</i> must be in a range from 0 to and including <code>COLOR_PAIRS-1</code>. Valid values for <i>f</i> and <i>b</i> are the range from 0 to and including <code>COLORS-1</code>.</p> | | | | | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>color</i></td><td>Is the number of the color for which to provide information (0 to <code>COLORS-1</code>).</td></tr><tr><td><i>red</i></td><td>Is a pointer to the RGB value for the amount of red in <i>color</i>.</td></tr><tr><td><i>green</i></td><td>Is a pointer to the RGB value for the amount of green in <i>color</i>.</td></tr><tr><td><i>blue</i></td><td>Is a pointer to the RGB value for the amount of blue in <i>color</i>.</td></tr><tr><td><i>n</i></td><td>Is the number of a color pair.</td></tr><tr><td><i>pair</i></td><td>Is the number of the color pair for which to provide information (1 to <code>COLOR_PAIRS-1</code>).</td></tr><tr><td><i>f</i></td><td>Is a pointer to the number of the foreground color (0 to <code>COLORS-1</code>) in <i>pair</i>.</td></tr></table> | <i>color</i> | Is the number of the color for which to provide information (0 to <code>COLORS-1</code>). | <i>red</i> | Is a pointer to the RGB value for the amount of red in <i>color</i> . | <i>green</i> | Is a pointer to the RGB value for the amount of green in <i>color</i> . | <i>blue</i> | Is a pointer to the RGB value for the amount of blue in <i>color</i> . | <i>n</i> | Is the number of a color pair. | <i>pair</i> | Is the number of the color pair for which to provide information (1 to <code>COLOR_PAIRS-1</code>). | <i>f</i> | Is a pointer to the number of the foreground color (0 to <code>COLORS-1</code>) in <i>pair</i> . |
| <i>color</i> | Is the number of the color for which to provide information (0 to <code>COLORS-1</code>). | | | | | | | | | | | | | | |
| <i>red</i> | Is a pointer to the RGB value for the amount of red in <i>color</i> . | | | | | | | | | | | | | | |
| <i>green</i> | Is a pointer to the RGB value for the amount of green in <i>color</i> . | | | | | | | | | | | | | | |
| <i>blue</i> | Is a pointer to the RGB value for the amount of blue in <i>color</i> . | | | | | | | | | | | | | | |
| <i>n</i> | Is the number of a color pair. | | | | | | | | | | | | | | |
| <i>pair</i> | Is the number of the color pair for which to provide information (1 to <code>COLOR_PAIRS-1</code>). | | | | | | | | | | | | | | |
| <i>f</i> | Is a pointer to the number of the foreground color (0 to <code>COLORS-1</code>) in <i>pair</i> . | | | | | | | | | | | | | | |

b Is a pointer to the number of the background color (0 to COLORS-1) in *pair*.

value Is a color attribute value.

RETURN VALUES The `has_colors()` function returns `TRUE` if the terminal can manipulate colors. Otherwise, it returns `FALSE`.

The `can_change_color()` function returns `TRUE` if the terminal supports colors and is able to change their definitions. Otherwise, it returns `FALSE`.

Upon successful completion, the other functions return `OK`. Otherwise, they return `ERR`.

ERRORS No errors are defined.

USAGE To use these functions, `start_color()` must be called, usually right after `initscr(3XCURSES)`.

The `can_change_color()` and `has_colors()` functions facilitate writing terminal-independent applications. For example, a programmer can use them to decide whether to use color or some other video attribute.

On color terminals, a typical value of `COLORS` is 8 and the macros such as `COLOR_BLACK` return a value within the range from 0 to and including 7. However, applications cannot rely on this to be true.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attroff(3XCURSES)`, `delscreen(3XCURSES)`, `initscr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

panel_above(3CURSES)

NAME panel_above, panel_below – panels deck traversal primitives

SYNOPSIS

```
cc [ flag ... ] file ... -lpanel -lcurses [ library .. ]
#include <panel.h>

PANEL *panel_above (PANEL *panel) ;
PANEL *panel_below (PANEL *panel) ;
```

DESCRIPTION panel_above () returns a pointer to the panel just above *panel*, or NULL if *panel* is the top panel. panel_below () returns a pointer to the panel just below *panel*, or NULL if *panel* is the bottom panel.

If NULL is passed for *panel*, panel_above () returns a pointer to the bottom panel in the deck, and panel_below () returns a pointer to the top panel in the deck.

RETURN VALUES NULL is returned if an error occurs.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), panels(3CURSES), attributes(5)

NOTES These routines allow traversal of the deck of currently visible panels.

The header <panel.h> automatically includes the header <curses.h>.

panel_below(3CURSES)

NAME panel_above, panel_below – panels deck traversal primitives

SYNOPSIS

```
cc [ flag ... ] file ... -lpanel -lcurses [ library .. ]
#include <panel.h>

PANEL *panel_above (PANEL *panel) ;
PANEL *panel_below (PANEL *panel) ;
```

DESCRIPTION panel_above () returns a pointer to the panel just above *panel*, or NULL if *panel* is the top panel. panel_below () returns a pointer to the panel just below *panel*, or NULL if *panel* is the bottom panel.

If NULL is passed for *panel*, panel_above () returns a pointer to the bottom panel in the deck, and panel_below () returns a pointer to the top panel in the deck.

RETURN VALUES NULL is returned if an error occurs.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), panels(3CURSES), attributes(5)

NOTES These routines allow traversal of the deck of currently visible panels.

The header <panel.h> automatically includes the header <curses.h>.

panel_hidden(3CURSES)

NAME panel_show, show_panel, hide_panel, panel_hidden – panels deck manipulation routines

SYNOPSIS

```
cc [ flag ... ] file ... -lpanel -lcurses [ library .. ]
#include <panel.h>

int show_panel (PANEL *panel) ;
int hide_panel (PANEL *panel) ;
int panel_hidden (PANEL *panel) ;
```

DESCRIPTION show_panel () makes *panel*, previously hidden, visible and places it on top of the deck of panels.

hide_panel () removes *panel* from the panel deck and, thus, hides it from view. The internal data structure of the panel is retained.

panel_hidden () returns TRUE (1) or FALSE (0) indicating whether or not *panel* is in the deck of panels.

RETURN VALUES show_panel () and hide_panel () return the integer OK upon successful completion or ERR upon error.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), panel_update(3CURSES), panels(3CURSES), attributes(5)

NOTES The header <panel.h> automatically includes the header <curses.h>.

NAME panel_move, move_panel – move a panels window on the virtual screen

SYNOPSIS `cc [flag ...] file ... -lpanel -lcurses [library ..]
#include <panel.h>`

```
int move_panel(PANEL *panel, int starty, int startx);
```

DESCRIPTION move_panel() moves the curses window associated with *panel* so that its upper left-hand corner is at *starty*, *startx*. See usage note, below.

RETURN VALUES OK is returned if the routine completes successfully, otherwise ERR is returned.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), panel_update(3CURSES), panels(3CURSES), attributes(5)

NOTES For panels windows, use move_panel() instead of the mvwin() curses routine. Otherwise, update_panels() will not properly update the virtual screen.

The header <panel.h> automatically includes the header <curses.h>.

panel_new(3CURSES)

NAME panel_new, new_panel, del_panel – create and destroy panels

SYNOPSIS `cc [flag ...] file ... -lpanel -lcurses [library ..]
#include <panel.h>`

```
PANEL *new_panel (WINDOW *win) ;  
int del_panel (PANEL *panel) ;
```

DESCRIPTION new_panel () creates a new panel associated with *win* and returns the panel pointer. The new panel is placed on top of the panel deck.

del_panel () destroys *panel*, but not its associated window.

RETURN VALUES new_panel () returns NULL if an error occurs.

del_win () returns OK if successful, ERR otherwise.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), panel_update(3CURSES), panels(3CURSES), attributes(5)

NOTES The header <panel.h> automatically includes the header <curses.h>.

| | |
|---------------------------|---|
| NAME | panels – character based panels package |
| SYNOPSIS | <code>#include <panel.h></code> |
| DESCRIPTION | <p>The panel library is built using the <code>curses</code> library, and any program using <code>panels</code> routines must call one of the <code>curses</code> initialization routines such as <code>initscr</code>. A program using these routines must be compiled with <code>-lpanel</code> and <code>-lcurses</code> on the <code>cc</code> command line.</p> <p>The <code>panels</code> package gives the applications programmer a way to have depth relationships between <code>curses</code> windows; a <code>curses</code> window is associated with every panel. The <code>panels</code> routines allow <code>curses</code> windows to overlap without making visible the overlapped portions of underlying windows. The initial <code>curses</code> window, <code>stdscr</code>, lies beneath all panels. The set of currently visible panels is the <i>deck</i> of panels.</p> <p>The <code>panels</code> package allows the applications programmer to create panels, fetch and set their associated windows, shuffle panels in the deck, and manipulate panels in other ways.</p> |
| Routine Name Index | <p>The following table lists each <code>panels</code> routine and the name of the manual page on which it is described.</p> |

| panels Routine Name | Manual Page Name |
|--------------------------------|-------------------------------------|
| <code>bottom_panel</code> | <code>panel_top(3CURSES)</code> |
| <code>del_panel</code> | <code>panel_new(3CURSES)</code> |
| <code>hide_panel</code> | <code>panel_show(3CURSES)</code> |
| <code>move_panel</code> | <code>panel_move(3CURSES)</code> |
| <code>new_panel</code> | <code>panel_new(3CURSES)</code> |
| <code>panel_above</code> | <code>panel_above(3CURSES)</code> |
| <code>panel_below</code> | <code>panel_above(3CURSES)</code> |
| <code>panel_hidden</code> | <code>panel_show(3CURSES)</code> |
| <code>panel_userptr</code> | <code>panel_userptr(3CURSES)</code> |
| <code>panel_window</code> | <code>panel_window(3CURSES)</code> |
| <code>replace_panel</code> | <code>panel_window(3CURSES)</code> |
| <code>set_panel_userptr</code> | <code>panel_userptr(3CURSES)</code> |
| <code>show_panel</code> | <code>panel_show(3CURSES)</code> |
| <code>top_panel</code> | <code>panel_top(3CURSES)</code> |
| <code>update_panels</code> | <code>panel_update(3CURSES)</code> |

panels(3CURSES)

RETURN VALUES Each `panels` routine that returns a pointer to an object returns `NULL` if an error occurs. Each `panel` routine that returns an integer, returns `OK` if it executes successfully and `ERR` if it does not.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3CURSES)`, `attributes(5)` and 3X pages whose names begin “`panel_`” for detailed routine descriptions.

NOTES The header `<panel.h>` automatically includes the header `<curses.h>`.

| NAME | panel_show, show_panel, hide_panel, panel_hidden – panels deck manipulation routines | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lpanel -lcurses [library ..] #include <panel.h> int show_panel (PANEL *panel) ; int hide_panel (PANEL *panel) ; int panel_hidden (PANEL *panel) ;</pre> | | | | |
| DESCRIPTION | <p>show_panel () makes <i>panel</i>, previously hidden, visible and places it on top of the deck of panels.</p> <p>hide_panel () removes <i>panel</i> from the panel deck and, thus, hides it from view. The internal data structure of the panel is retained.</p> <p>panel_hidden () returns TRUE (1) or FALSE (0) indicating whether or not <i>panel</i> is in the deck of panels.</p> | | | | |
| RETURN VALUES | show_panel () and hide_panel () return the integer OK upon successful completion or ERR upon error. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), panel_update(3CURSES), panels(3CURSES), attributes(5) | | | | |
| NOTES | The header <panel.h> automatically includes the header <curses.h>. | | | | |

panel_top(3CURSES)

NAME panel_top, top_panel, bottom_panel – panels deck manipulation routines

SYNOPSIS

```
cc [ flag ... ] file ... -lpanel -lcurses [ library .. ]
#include <panel.h>

int top_panel (PANEL *panel) ;
int bottom_panel (PANEL *panel) ;
```

DESCRIPTION top_panel () pulls *panel* to the top of the desk of panels. It leaves the size, location, and contents of its associated window unchanged.

bottom_panel () puts *panel* at the bottom of the deck of panels. It leaves the size, location, and contents of its associated window unchanged.

RETURN VALUES All of these routines return the integer OK upon successful completion or ERR upon error.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), panel_update(3CURSES), panels(3CURSES), attributes(5)

NOTES The header <panel.h> automatically includes the header <curses.h>.

panel_update(3CURSES)

NAME panel_update, update_panels – panels virtual screen refresh routine

SYNOPSIS

```
cc [ flag ... ] file ... -lpanel -lcurses [ library .. ]
#include <panel.h>

void update_panels(void);
```

DESCRIPTION update_panels() refreshes the virtual screen to reflect the depth relationships between the panels in the deck. The user must use the curses library call doupdate() (see curs_refresh(3CURSES)) to refresh the physical screen.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_refresh(3CURSES), curses(3CURSES), panels(3CURSES), attributes(5)

NOTES The header <panel.h> automatically includes the header <curses.h>.

panel_userptr(3CURSES)

NAME panel_userptr, set_panel_userptr – associate application data with a panels panel

SYNOPSIS

```
cc [ flag ... ] file ... -lpanel -lcurses [ library .. ]
#include <panel.h>

int set_panel_userptr(PANEL *panel, char *ptr);
char * panel_userptr(PANEL *panel);
```

DESCRIPTION Each panel has a user pointer available for maintaining relevant information.

set_panel_userptr() sets the user pointer of *panel* to *ptr*.

panel_userptr() returns the user pointer of *panel*.

RETURN VALUES set_panel_userptr returns OK if successful, ERR otherwise.

panel_userptr returns NULL if there is no user pointer assigned to *panel*.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), panels(3CURSES), attributes(5)

NOTES The header <panel.h> automatically includes the header <curses.h>.

NAME panel_window, replace_panel – get or set the current window of a panels panel

SYNOPSIS

```
cc [ flag ... ] file ... -lpanel -lcurses [ library .. ]
#include <panel.h>

WINDOW *panel_window(PANEL *panel);
int replace_panel(PANEL *panel, WINDOW *win);
```

DESCRIPTION panel_window() returns a pointer to the window of *panel*.
replace_panel() replaces the current window of *panel* with *win*.

RETURN VALUES panel_window() returns NULL on failure.
replace_panel() returns OK on successful completion, ERR otherwise.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), panels(3CURSES), attributes(5)

NOTES The header <panel.h> automatically includes the header <curses.h>.

pechochar(3CURSES)

| | |
|--------------------|--|
| NAME | <code>curs_pad</code> , <code>newpad</code> , <code>subpad</code> , <code>prefresh</code> , <code>pnoutrefresh</code> , <code>pechochar</code> , <code>pechowchar</code> – create and display curses pads |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> WINDOW *newpad(int nlines, int ncols); WINDOW *subpad(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); int prefresh(WINDOW *pad, int pminrow, int pmincol, int sminrow, int smincol, int smaxrow, int smaxcol); int pnoutrefresh(WINDOW *pad, int pminrow, int pmincol, int sminrow, int smincol, int smaxrow, int smaxcol); int pechochar(WINDOW *pad, chtype ch); int pechowchar(WINDOW *pad, chtype wch);</pre> |
| DESCRIPTION | <p>The <code>newpad()</code> routine creates and returns a pointer to a new pad data structure with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. A pad is like a window, except that it is not restricted by the screen size, and is not necessarily associated with a particular part of the screen. Pads can be used when a large window is needed, and only a part of the window will be on the screen at one time. Automatic refreshes of pads (for example, from scrolling or echoing of input) do not occur. It is not legal to call <code>wrefresh(3CURSES)</code> with a <i>pad</i> as an argument; the routines <code>prefresh()</code> or <code>pnoutrefresh()</code> should be called instead. Note that these routines require additional parameters to specify the part of the pad to be displayed and the location on the screen to be used for the display.</p> <p>The <code>subpad()</code> routine creates and returns a pointer to a subwindow within a pad with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. Unlike <code>subwin(3CURSES)</code>, which uses screen coordinates, the window is at position (<i>begin_x</i>, <i>begin_y</i>) on the pad. The window is made in the middle of the window <i>orig</i>, so that changes made to one window affect both windows. During the use of this routine, it will often be necessary to call <code>touchwin(3CURSES)</code> or <code>touchline(3CURSES)</code> on <i>orig</i> before calling <code>prefresh()</code>.</p> <p>The <code>prefresh()</code> and <code>pnoutrefresh()</code> routines are analogous to <code>wrefresh(3CURSES)</code> and <code>wnoutrefresh(3CURSES)</code> except that they relate to pads instead of windows. The additional parameters are needed to indicate what part of the pad and screen are involved. <i>pminrow</i> and <i>pmincol</i> specify the upper left-hand corner of the rectangle to be displayed in the pad. <i>sminrow</i>, <i>smincol</i>, <i>smaxrow</i>, and <i>smaxcol</i> specify the edges of the rectangle to be displayed on the screen. The lower right-hand corner of the rectangle to be displayed in the pad is calculated from the screen coordinates, since the rectangles must be the same size. Both rectangles must be entirely contained within their respective structures. Negative values of <i>pminrow</i>, <i>pmincol</i>, <i>sminrow</i>, or <i>smincol</i> are treated as if they were zero.</p> |

pechochar(3CURSES)

The `pechochar()` routine is functionally equivalent to a call to `addch(3CURSES)` followed by a call to `refresh(3CURSES)`, a call to `waddch(3CURSES)` followed by a call to `wrefresh(3CURSES)`, or a call to `waddch(3CURSES)` followed by a call to `prefresh()`. The knowledge that only a single character is being output is taken into consideration and, for non-control characters, a considerable performance gain might be seen by using these routines instead of their equivalents. In the case of `pechochar()`, the last location of the pad on the screen is reused for the arguments to `prefresh()`.

RETURN VALUES Routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

Routines that return pointers return `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `addch(3CURSES)`, `curses(3CURSES)`, `refresh(3CURSES)`, `subwin(3CURSES)`, `touchline(3CURSES)`, `touchwin(3CURSES)`, `waddch(3CURSES)`, `wnoutrefresh(3CURSES)`, `wrefresh(3CURSES)`, `attributes(5)`

NOTES The header file `<curses.h>` automatically includes the header files `<stdio.h>`, `<unctrl.h>` and `<widec.h>`.

Note that `pechochar()` may be a macro.

pechochar(3XCURSES)

| NAME | pechochar, pecho_wchar – add character and refresh window | | | | | | |
|----------------------|---|----------------|--|---------------------|---|------------|---|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int pechochar(WINDOW *pad, chtype ch); int pecho_wchar(WINDOW *pad, const chtype *wch);</pre> | | | | | | |
| PARAMETERS | <table><tr><td><i>pad</i></td><td>Is a pointer to the pad in which the character is to be added.</td></tr><tr><td><i>ch</i></td><td>Is a pointer to the character to be written to the pad.</td></tr><tr><td><i>wch</i></td><td>Is a pointer to the complex character to be written to the pad.</td></tr></table> | <i>pad</i> | Is a pointer to the pad in which the character is to be added. | <i>ch</i> | Is a pointer to the character to be written to the pad. | <i>wch</i> | Is a pointer to the complex character to be written to the pad. |
| <i>pad</i> | Is a pointer to the pad in which the character is to be added. | | | | | | |
| <i>ch</i> | Is a pointer to the character to be written to the pad. | | | | | | |
| <i>wch</i> | Is a pointer to the complex character to be written to the pad. | | | | | | |
| DESCRIPTION | The pechochar() function is equivalent to calling waddch(3XCURSES) followed by a call to prefresh(3XCURSES). The pecho_wchar() function is equivalent to calling wadd_wch(3XCURSES) followed by a call to prefresh(). prefresh() reuses the last position of the pad on the screen for its parameters. | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Standard</td></tr><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | add_wch(3XCURSES), addch(3XCURSES), libcurses(3XCURSES), newpad(3XCURSES), attributes(5), standards(5) | | | | | | |

| | |
|--------------------|--|
| NAME | curs_pad, newpad, subpad, prefresh, pnoutrefresh, pechochar, pechowchar – create and display curses pads |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> WINDOW *newpad(int nlines, int ncols); WINDOW *subpad(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); int prefresh(WINDOW *pad, int pminrow, int pmincol, int sminrow, int smincol, int smaxrow, int smaxcol); int pnoutrefresh(WINDOW *pad, int pminrow, int pmincol, int sminrow, int smincol, int smaxrow, int smaxcol); int pechochar(WINDOW *pad, chtype ch); int pechowchar(WINDOW *pad, chtype wch);</pre> |
| DESCRIPTION | <p>The <code>newpad()</code> routine creates and returns a pointer to a new pad data structure with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. A pad is like a window, except that it is not restricted by the screen size, and is not necessarily associated with a particular part of the screen. Pads can be used when a large window is needed, and only a part of the window will be on the screen at one time. Automatic refreshes of pads (for example, from scrolling or echoing of input) do not occur. It is not legal to call <code>wrefresh(3CURSES)</code> with a <i>pad</i> as an argument; the routines <code>prefresh()</code> or <code>pnoutrefresh()</code> should be called instead. Note that these routines require additional parameters to specify the part of the pad to be displayed and the location on the screen to be used for the display.</p> <p>The <code>subpad()</code> routine creates and returns a pointer to a subwindow within a pad with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. Unlike <code>subwin(3CURSES)</code>, which uses screen coordinates, the window is at position (<i>begin_x</i>, <i>begin_y</i>) on the pad. The window is made in the middle of the window <i>orig</i>, so that changes made to one window affect both windows. During the use of this routine, it will often be necessary to call <code>touchwin(3CURSES)</code> or <code>touchline(3CURSES)</code> on <i>orig</i> before calling <code>prefresh()</code>.</p> <p>The <code>prefresh()</code> and <code>pnoutrefresh()</code> routines are analogous to <code>wrefresh(3CURSES)</code> and <code>wnoutrefresh(3CURSES)</code> except that they relate to pads instead of windows. The additional parameters are needed to indicate what part of the pad and screen are involved. <i>pminrow</i> and <i>pmincol</i> specify the upper left-hand corner of the rectangle to be displayed in the pad. <i>sminrow</i>, <i>smincol</i>, <i>smaxrow</i>, and <i>smaxcol</i> specify the edges of the rectangle to be displayed on the screen. The lower right-hand corner of the rectangle to be displayed in the pad is calculated from the screen coordinates, since the rectangles must be the same size. Both rectangles must be entirely contained within their respective structures. Negative values of <i>pminrow</i>, <i>pmincol</i>, <i>sminrow</i>, or <i>smincol</i> are treated as if they were zero.</p> |

pechochar(3CURSES)

The pechochar() routine is functionally equivalent to a call to addch(3CURSES) followed by a call to refresh(3CURSES), a call to waddch(3CURSES) followed by a call to wrefresh(3CURSES), or a call to waddch(3CURSES) followed by a call to prefresh(). The knowledge that only a single character is being output is taken into consideration and, for non-control characters, a considerable performance gain might be seen by using these routines instead of their equivalents. In the case of pechochar(), the last location of the pad on the screen is reused for the arguments to prefresh().

RETURN VALUES Routines that return an integer return ERR upon failure and an integer value other than ERR upon successful completion.

Routines that return pointers return NULL on error.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO addch(3CURSES), curses(3CURSES), refresh(3CURSES), subwin(3CURSES), touchline(3CURSES), touchwin(3CURSES), waddch(3CURSES), wnoutrefresh(3CURSES), wrefresh(3CURSES), attributes(5)

NOTES The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <wdec.h>.

Note that pechochar() may be a macro.

| NAME | pechochar, pecho_wchar – add character and refresh window | | | | | | |
|----------------------|--|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int pechochar(WINDOW *pad, chtype ch); int pecho_wchar(WINDOW *pad, const chtype *wch);</pre> | | | | | | |
| PARAMETERS | <p><i>pad</i> Is a pointer to the pad in which the character is to be added.</p> <p><i>ch</i> Is a pointer to the character to be written to the pad.</p> <p><i>wch</i> Is a pointer to the complex character to be written to the pad.</p> | | | | | | |
| DESCRIPTION | The pechochar() function is equivalent to calling waddch(3XCURSES) followed by a call to prefresh(3XCURSES). The pecho_wchar() function is equivalent to calling wadd_wch(3XCURSES) followed by a call to prefresh(). prefresh() reuses the last position of the pad on the screen for its parameters. | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | add_wch(3XCURSES), addch(3XCURSES), libcurses(3XCURSES), newpad(3XCURSES), attributes(5), standards(5) | | | | | | |

plot(3PLOT)

| | |
|--------------------|--|
| NAME | plot, arc, box, circle, closepl, closevt, cont, erase, label, line, linemod, move, openpl, openvt, point, space – graphics interface |
| SYNOPSIS | <pre>cc [flag ...] file ... -lplot [library...] #include <plot.h> void arc(short x0, short y0, short x1, short y1, short x2, short y2); void box(short x0, short y0, short x1, short y1); void circle(short x, short y, short r); void closepl(); void closevt(); void cont(short x, short y); void erase(); void label(char *s); void line(short x0, short y0, short x1, short y1); void linemod(char *s); void move(short x, short y); void openpl(); void openvt(); void point(short x, short y); void space(short x0, short y0, short x1, short y1);</pre> |
| DESCRIPTION | <p>These functions generate graphics output for a set of output devices. The format of the output is dependent upon which link editor option is used when the program is compiled and linked (see Link Editor).</p> <p>The term "current point" refers to the current setting for the x and y coordinates.</p> <p>The <code>arc()</code> function specifies a circular arc. The coordinates $(x0, y0)$ specify the center of the arc. The coordinates $(x1, y1)$ specify the starting point of the arc. The coordinates $(x2, y2)$ specify the end point of the circular arc.</p> <p>The <code>box()</code> function specifies a rectangle with coordinates $(x0, y0)$, $(x0, y1)$, $(x1, y0)$, and $(x1, y1)$. The current point is set to $(x1, y1)$.</p> <p>The <code>circle()</code> function specifies a circle with a center at the coordinates (x, y) and a radius of r.</p> <p>The <code>closevt()</code> and <code>closepl()</code> functions flush the output.</p> <p>The <code>cont()</code> function specifies a line beginning at the current point and ending at the coordinates (x, y). The current point is set to (x, y).</p> |

The `erase()` function starts another frame of output.

The `label()` function places the null terminated string `s` so that the first character falls on the current point. The string is then terminated by a NEWLINE character.

The `line()` function draws a line starting at the coordinates $(x0, y0)$ and ending at the coordinates $(x1, y1)$. The current point is set to $(x1, y1)$.

The `linemod()` function specifies the style for drawing future lines. `s` may contain one of the following: `dotted`, `solid`, `longdashed`, `shortdashed`, or `dotdashed`.

The `move()` function sets the current point to the coordinates (x, y) .

The `openpl()` or `openvt()` function must be called to open the device before any other `plot` functions are called.

The `point()` function plots the point given by the coordinates (x, y) . The current point is set to (x, y) .

The `space()` function specifies the size of the plotting area. The plot will be reduced or enlarged as necessary to fit the area specified. The coordinates $(x0, y0)$ specify the lower left hand corner of the plotting area. The coordinates $(x1, y1)$ specify the upper right hand corner of the plotting area.

Link Editor Various flavors of these functions exist for different output devices. They are obtained by using the following `ld(1)` options:

```
-lplot          device-independent graphics stream on standard output in the
                 format described in plot(4B)
-l300           GSI 300 terminal
-l300s          GSI 300S terminal
-l4014          Tektronix 4014 terminal
-l450           GSI 450 terminal
-lvt0
```

FILES

```
/usr/lib/libplot.a
  archive library

/usr/lib/libplot.so.1
  shared object

/usr/lib/sparcv9/libplot.so.1
  64-bit shared object

/usr/lib/lib300.a
  archive library

/usr/lib/lib300.so.1
  shared object
```

plot(3PLOT)

/usr/lib/sparcv9/lib300.so.1
64-bit shared object

/usr/lib/lib300s.a
archive library

/usr/lib/lib300s.so.1
shared object

/usr/lib/sparcv9/lib300s.so.1
64-bit shared object

/usr/lib/lib4014.a
archive library

/usr/lib/lib4014.so.1
shared object

/usr/lib/sparcv9/lib4014.so.1
64-bit shared object

/usr/lib/lib450.a
archive library

/usr/lib/lib450.so.1
shared object

/usr/lib/sparcv9/lib450.so.1
64-bit shared object

/usr/lib/libvt0.a
archive library

/usr/lib/libvt0.so.1
shared object

/usr/lib/sparcv9/libvt0.so.1
64-bit shared object

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `graph(1)`, `ld(1)`, `libplot(3LIB)`, `plot(4B)`, `attributes(5)`

| | |
|--------------------|--|
| NAME | curs_pad, newpad, subpad, prefresh, pnoutrefresh, pechochar, pechowchar – create and display curses pads |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> WINDOW *newpad(int nlines, int ncols); WINDOW *subpad(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); int prefresh(WINDOW *pad, int pminrow, int pmincol, int sminrow, int smincol, int smaxrow, int smaxcol); int pnoutrefresh(WINDOW *pad, int pminrow, int pmincol, int sminrow, int smincol, int smaxrow, int smaxcol); int pechochar(WINDOW *pad, chtype ch); int pechowchar(WINDOW *pad, chtype wch);</pre> |
| DESCRIPTION | <p>The <code>newpad()</code> routine creates and returns a pointer to a new pad data structure with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. A pad is like a window, except that it is not restricted by the screen size, and is not necessarily associated with a particular part of the screen. Pads can be used when a large window is needed, and only a part of the window will be on the screen at one time. Automatic refreshes of pads (for example, from scrolling or echoing of input) do not occur. It is not legal to call <code>wrefresh(3CURSES)</code> with a <i>pad</i> as an argument; the routines <code>prefresh()</code> or <code>pnoutrefresh()</code> should be called instead. Note that these routines require additional parameters to specify the part of the pad to be displayed and the location on the screen to be used for the display.</p> <p>The <code>subpad()</code> routine creates and returns a pointer to a subwindow within a pad with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. Unlike <code>subwin(3CURSES)</code>, which uses screen coordinates, the window is at position (<i>begin_x</i>, <i>begin_y</i>) on the pad. The window is made in the middle of the window <i>orig</i>, so that changes made to one window affect both windows. During the use of this routine, it will often be necessary to call <code>touchwin(3CURSES)</code> or <code>touchline(3CURSES)</code> on <i>orig</i> before calling <code>prefresh()</code>.</p> <p>The <code>prefresh()</code> and <code>pnoutrefresh()</code> routines are analogous to <code>wrefresh(3CURSES)</code> and <code>wnoutrefresh(3CURSES)</code> except that they relate to pads instead of windows. The additional parameters are needed to indicate what part of the pad and screen are involved. <i>pminrow</i> and <i>pmincol</i> specify the upper left-hand corner of the rectangle to be displayed in the pad. <i>sminrow</i>, <i>smincol</i>, <i>smaxrow</i>, and <i>smaxcol</i> specify the edges of the rectangle to be displayed on the screen. The lower right-hand corner of the rectangle to be displayed in the pad is calculated from the screen coordinates, since the rectangles must be the same size. Both rectangles must be entirely contained within their respective structures. Negative values of <i>pminrow</i>, <i>pmincol</i>, <i>sminrow</i>, or <i>smincol</i> are treated as if they were zero.</p> |

pnoutrefresh(3CURSES)

The `pechochar()` routine is functionally equivalent to a call to `addch(3CURSES)` followed by a call to `refresh(3CURSES)`, a call to `waddch(3CURSES)` followed by a call to `wrefresh(3CURSES)`, or a call to `waddch(3CURSES)` followed by a call to `prefresh()`. The knowledge that only a single character is being output is taken into consideration and, for non-control characters, a considerable performance gain might be seen by using these routines instead of their equivalents. In the case of `pechochar()`, the last location of the pad on the screen is reused for the arguments to `prefresh()`.

RETURN VALUES Routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

Routines that return pointers return `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `addch(3CURSES)`, `curses(3CURSES)`, `refresh(3CURSES)`, `subwin(3CURSES)`, `touchline(3CURSES)`, `touchwin(3CURSES)`, `waddch(3CURSES)`, `wnoutrefresh(3CURSES)`, `wrefresh(3CURSES)`, `attributes(5)`

NOTES The header file `<curses.h>` automatically includes the header files `<stdio.h>`, `<unctrl.h>` and `<widec.h>`.

Note that `pechochar()` may be a macro.

| | | | | | | | | | | | | | | | | | | | | | |
|--------------------|--|---------------|--|--------------|--|------------|-------------------------------------|----------------|--|----------------|--|----------------|---|----------------|--|----------------|--|----------------|---|-------------|---|
| NAME | newpad, pnoutrefresh, prefresh, subpad – create or refresh a pad or subpad | | | | | | | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> WINDOW *newpad(int nlines, int ncols); int pnoutrefresh(WINDOW *pad, int pminrow, int pmincol, int sminrow, int smincol, int smaxrow, int smaxcol); int prefresh(WINDOW *pad, int pminrow, int pmincol, int sminrow, int smincol, int smaxrow, int smaxcol); WINDOW *subpad(WINDOW *orig, int nlines, int ncols);</pre> | | | | | | | | | | | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>nlines</i></td> <td>Is the number of lines in the pad to be created.</td> </tr> <tr> <td><i>ncols</i></td> <td>Is the number of columns in the pad to be created.</td> </tr> <tr> <td><i>pad</i></td> <td>Is a pointer to the pad to refresh.</td> </tr> <tr> <td><i>pminrow</i></td> <td>Is the row coordinate of the upper left corner of the pad rectangle to be copied</td> </tr> <tr> <td><i>pmincol</i></td> <td>Is the column coordinate of the upper left corner of the pad rectangle to be copied.</td> </tr> <tr> <td><i>sminrow</i></td> <td>Is the row coordinate of the upper left corner of the rectangle on the physical screen where pad is to be positioned.</td> </tr> <tr> <td><i>smincol</i></td> <td>Is the column coordinate of the upper left corner of the rectangle on the physical screen where pad is to be positioned.</td> </tr> <tr> <td><i>smaxrow</i></td> <td>Is the row coordinate of the lower right corner of the rectangle on the physical screen where the pad is to be positioned.</td> </tr> <tr> <td><i>smaxcol</i></td> <td>Is the column coordinate of the lower right corner of the rectangle on the physical screen where the pad is to be positioned.</td> </tr> <tr> <td><i>orig</i></td> <td>Is a pointer to the parent pad within which a sub-pad is created.</td> </tr> </table> | <i>nlines</i> | Is the number of lines in the pad to be created. | <i>ncols</i> | Is the number of columns in the pad to be created. | <i>pad</i> | Is a pointer to the pad to refresh. | <i>pminrow</i> | Is the row coordinate of the upper left corner of the pad rectangle to be copied | <i>pmincol</i> | Is the column coordinate of the upper left corner of the pad rectangle to be copied. | <i>sminrow</i> | Is the row coordinate of the upper left corner of the rectangle on the physical screen where pad is to be positioned. | <i>smincol</i> | Is the column coordinate of the upper left corner of the rectangle on the physical screen where pad is to be positioned. | <i>smaxrow</i> | Is the row coordinate of the lower right corner of the rectangle on the physical screen where the pad is to be positioned. | <i>smaxcol</i> | Is the column coordinate of the lower right corner of the rectangle on the physical screen where the pad is to be positioned. | <i>orig</i> | Is a pointer to the parent pad within which a sub-pad is created. |
| <i>nlines</i> | Is the number of lines in the pad to be created. | | | | | | | | | | | | | | | | | | | | |
| <i>ncols</i> | Is the number of columns in the pad to be created. | | | | | | | | | | | | | | | | | | | | |
| <i>pad</i> | Is a pointer to the pad to refresh. | | | | | | | | | | | | | | | | | | | | |
| <i>pminrow</i> | Is the row coordinate of the upper left corner of the pad rectangle to be copied | | | | | | | | | | | | | | | | | | | | |
| <i>pmincol</i> | Is the column coordinate of the upper left corner of the pad rectangle to be copied. | | | | | | | | | | | | | | | | | | | | |
| <i>sminrow</i> | Is the row coordinate of the upper left corner of the rectangle on the physical screen where pad is to be positioned. | | | | | | | | | | | | | | | | | | | | |
| <i>smincol</i> | Is the column coordinate of the upper left corner of the rectangle on the physical screen where pad is to be positioned. | | | | | | | | | | | | | | | | | | | | |
| <i>smaxrow</i> | Is the row coordinate of the lower right corner of the rectangle on the physical screen where the pad is to be positioned. | | | | | | | | | | | | | | | | | | | | |
| <i>smaxcol</i> | Is the column coordinate of the lower right corner of the rectangle on the physical screen where the pad is to be positioned. | | | | | | | | | | | | | | | | | | | | |
| <i>orig</i> | Is a pointer to the parent pad within which a sub-pad is created. | | | | | | | | | | | | | | | | | | | | |
| DESCRIPTION | The newpad() function creates a new pad with the specified number of lines and columns. A pointer to the new pad structure is returned. A pad differs from a window in that it is not restricted to the size of the physical screen. It is useful when only part of a large window will be displayed at any one time. | | | | | | | | | | | | | | | | | | | | |

pnoutrefresh(3XCURSES)

Automatic refreshes by scrolling or echoing of input do not take place when pads are used. Pads have their own refresh commands, `prefresh()` and `pnoutrefresh()`.

The `prefresh()` function copies the specified portion of the logical pad to the terminal screen. The parameters `pmincol` and `pminrow` specify the upper left corner of the rectangular area of the pad to be displayed. The lower right coordinate of the rectangular area of the pad that is to be displayed is calculated from the screen parameters (`sminrow`, `smincol`, `smaxrow`, `smaxcol`).

This function calls the `pnoutrefresh()` function to copy the specified portion of *pad* to the terminal screen and the `doupdate(3XCURSES)` function to do the actual update. The logical cursor is copied to the same location in the physical window unless `leaveok(3XCURSES)` is enabled (in which case, the cursor is placed in a position that the program finds convenient).

When outputting several pads at once, it is often more efficient to call the `pnoutrefresh()` and `doupdate()` functions directly. A call to `pnoutrefresh()` for each pad first, followed by only one call to `doupdate()` to update the screen, results in one burst of output, fewer characters sent, and less CPU time used.

The `subpad()` function creates a sub-pad within the pad *orig* with the specified number of lines and columns. A pointer to the new pad structure is returned. The sub-pad is positioned in the middle of *orig*. Any changes made to one pad affect the other. `touchwin(3XCURSES)` or `touchline(3XCURSES)` will likely have to be called on pad *orig* to correctly update the window.

RETURN VALUES

On success, the `newpad()` and `subpad()` functions return a pointer to the new pad data structure. Otherwise, they return a null pointer.

On success, the `pnoutrefresh()` and `prefresh()` functions return `OK`. Otherwise, they return `ERR`.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO

`clearok(3XCURSES)`, `doupdate(3XCURSES)`, `is_linetouched(3XCURSES)`, `libcurses(3XCURSES)`, `pechochar(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | plot, arc, box, circle, closepl, closevt, cont, erase, label, line, linemod, move, openpl, openvt, point, space – graphics interface |
| SYNOPSIS | <pre>cc [flag ...] file ... -lplot [library...] #include <plot.h> void arc(short x0, short y0, short x1, short y1, short x2, short y2); void box(short x0, short y0, short x1, short y1); void circle(short x, short y, short r); void closepl(); void closevt(); void cont(short x, short y); void erase(); void label(char *s); void line(short x0, short y0, short x1, short y1); void linemod(char *s); void move(short x, short y); void openpl(); void openvt(); void point(short x, short y); void space(short x0, short y0, short x1, short y1);</pre> |
| DESCRIPTION | <p>These functions generate graphics output for a set of output devices. The format of the output is dependent upon which link editor option is used when the program is compiled and linked (see Link Editor).</p> <p>The term "current point" refers to the current setting for the x and y coordinates.</p> <p>The <code>arc()</code> function specifies a circular arc. The coordinates $(x0, y0)$ specify the center of the arc. The coordinates $(x1, y1)$ specify the starting point of the arc. The coordinates $(x2, y2)$ specify the end point of the circular arc.</p> <p>The <code>box()</code> function specifies a rectangle with coordinates $(x0, y0)$, $(x0, y1)$, $(x1, y0)$, and $(x1, y1)$. The current point is set to $(x1, y1)$.</p> <p>The <code>circle()</code> function specifies a circle with a center at the coordinates (x, y) and a radius of r.</p> <p>The <code>closevt()</code> and <code>closepl()</code> functions flush the output.</p> <p>The <code>cont()</code> function specifies a line beginning at the current point and ending at the coordinates (x, y). The current point is set to (x, y).</p> |

point(3PLOT)

The `erase()` function starts another frame of output.

The `label()` function places the null terminated string `s` so that the first character falls on the current point. The string is then terminated by a NEWLINE character.

The `line()` function draws a line starting at the coordinates (x_0, y_0) and ending at the coordinates (x_1, y_1) . The current point is set to (x_1, y_1) .

The `linemod()` function specifies the style for drawing future lines. `s` may contain one of the following: `dotted`, `solid`, `longdashed`, `shortdashed`, or `dotdashed`.

The `move()` function sets the current point to the coordinates (x, y) .

The `openpl()` or `openvt()` function must be called to open the device before any other plot functions are called.

The `point()` function plots the point given by the coordinates (x, y) . The current point is set to (x, y) .

The `space()` function specifies the size of the plotting area. The plot will be reduced or enlarged as necessary to fit the area specified. The coordinates (x_0, y_0) specify the lower left hand corner of the plotting area. The coordinates (x_1, y_1) specify the upper right hand corner of the plotting area.

Link Editor

Various flavors of these functions exist for different output devices. They are obtained by using the following `ld(1)` options:

| | |
|---------------------|--|
| <code>-lplot</code> | device-independent graphics stream on standard output in the format described in <code>plot(4B)</code> |
| <code>-l300</code> | GSI 300 terminal |
| <code>-l300s</code> | GSI 300S terminal |
| <code>-l4014</code> | Tektronix 4014 terminal |
| <code>-l450</code> | GSI 450 terminal |
| <code>-lvt0</code> | |

FILES

| | |
|--|----------------------|
| <code>/usr/lib/libplot.a</code> | archive library |
| <code>/usr/lib/libplot.so.1</code> | shared object |
| <code>/usr/lib/sparcv9/libplot.so.1</code> | 64-bit shared object |
| <code>/usr/lib/lib300.a</code> | archive library |
| <code>/usr/lib/lib300.so.1</code> | shared object |

point(3PLOT)

/usr/lib/sparcv9/lib300.so.1
64-bit shared object
/usr/lib/lib300s.a
archive library
/usr/lib/lib300s.so.1
shared object
/usr/lib/sparcv9/lib300s.so.1
64-bit shared object
/usr/lib/lib4014.a
archive library
/usr/lib/lib4014.so.1
shared object
/usr/lib/sparcv9/lib4014.so.1
64-bit shared object
/usr/lib/lib450.a
archive library
/usr/lib/lib450.so.1
shared object
/usr/lib/sparcv9/lib450.so.1
64-bit shared object
/usr/lib/libvt0.a
archive library
/usr/lib/libvt0.so.1
shared object
/usr/lib/sparcv9/libvt0.so.1
64-bit shared object

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO graph(1), ld(1), libplot(3LIB), plot(4B), attributes(5)

pos_menu_cursor(3CURSES)

NAME menu_cursor, pos_menu_cursor – correctly position a menu cursor

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

int pos_menu_cursor(MENU *menu);
```

DESCRIPTION pos_menu_cursor() moves the cursor in the window of *menu* to the correct position to resume menu processing. This is needed after the application calls a curses library I/O routine.

RETURN VALUES This routine returns one of the following:

| | |
|----------------|--|
| E_OK | The routine returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. |
| E_NOT_POSTED | The menu has not been posted. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), panel_update(3CURSES), panels(3CURSES), attributes(5)

NOTES The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

post_form(3CURSES)

NAME form_post, post_form, unpost_form – write or erase forms from associated subwindows

SYNOPSIS

```
cc [ flag... ] file... -lform -lcurses [ library... ]
#include <form.h>

int post_form(FORM *form) ;
int unpost_form(FORM *form) ;
```

DESCRIPTION post_form() writes *form* into its associated subwindow. The application programmer must use curses library routines to display the form on the physical screen or call update_panels() if the panels library is being used.

unpost_form() erases *form* from its associated subwindow.

RETURN VALUES These routines return one of the following:

| | |
|-----------------|--|
| E_OK | The function returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An argument is incorrect. |
| E_POSTED | The form is posted. |
| E_NOT_POSTED | The form is not posted. |
| E_NO_ROOM | The form does not fit in the subwindow. |
| E_BAD_STATE | The routine was called from an initialization or termination function. |
| E_NOT_CONNECTED | The field is not connected to a form. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), forms(3CURSES), panel_update(3CURSES), panels(3CURSES), attributes(5)

NOTES The header <form.h> automatically includes the headers <eti.h> and <curses.h>.

NAME menu_post, post_menu, unpost_menu – write or erase menus from associated subwindows

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

int post_menu(MENU *menu);
int unpost_menu(MENU *menu);
```

DESCRIPTION post_menu() writes *menu* to the subwindow. The application programmer must use curses library routines to display the menu on the physical screen or call update_panels() if the panels library is being used.

unpost_menu() erases *menu* from its associated subwindow.

RETURN VALUES These routines return one of the following:

| | |
|-----------------|--|
| E_OK | The routine returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. |
| E_POSTED | The menu is already posted. |
| E_BAD_STATE | The routine was called from an initialization or termination function. |
| E_NO_ROOM | The menu does not fit within its subwindow. |
| E_NOT_POSTED | The menu has not been posted. |
| E_NOT_CONNECTED | No items are connected to the menu. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), panels(3CURSES), attributes(5)

NOTES The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

prefresh(3CURSES)

| | |
|--------------------|--|
| NAME | <code>curs_pad</code> , <code>newpad</code> , <code>subpad</code> , <code>prefresh</code> , <code>pnoutrefresh</code> , <code>pechochar</code> , <code>pechowchar</code> – create and display curses pads |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> WINDOW *newpad(int nlines, int ncols); WINDOW *subpad(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); int prefresh(WINDOW *pad, int pminrow, int pmincol, int sminrow, int smincol, int smaxrow, int smaxcol); int pnoutrefresh(WINDOW *pad, int pminrow, int pmincol, int sminrow, int smincol, int smaxrow, int smaxcol); int pechochar(WINDOW *pad, chtype ch); int pechowchar(WINDOW *pad, chtype wch);</pre> |
| DESCRIPTION | <p>The <code>newpad()</code> routine creates and returns a pointer to a new pad data structure with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. A pad is like a window, except that it is not restricted by the screen size, and is not necessarily associated with a particular part of the screen. Pads can be used when a large window is needed, and only a part of the window will be on the screen at one time. Automatic refreshes of pads (for example, from scrolling or echoing of input) do not occur. It is not legal to call <code>wrefresh(3CURSES)</code> with a <i>pad</i> as an argument; the routines <code>prefresh()</code> or <code>pnoutrefresh()</code> should be called instead. Note that these routines require additional parameters to specify the part of the pad to be displayed and the location on the screen to be used for the display.</p> <p>The <code>subpad()</code> routine creates and returns a pointer to a subwindow within a pad with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. Unlike <code>subwin(3CURSES)</code>, which uses screen coordinates, the window is at position (<i>begin_x</i>, <i>begin_y</i>) on the pad. The window is made in the middle of the window <i>orig</i>, so that changes made to one window affect both windows. During the use of this routine, it will often be necessary to call <code>touchwin(3CURSES)</code> or <code>touchline(3CURSES)</code> on <i>orig</i> before calling <code>prefresh()</code>.</p> <p>The <code>prefresh()</code> and <code>pnoutrefresh()</code> routines are analogous to <code>wrefresh(3CURSES)</code> and <code>wnoutrefresh(3CURSES)</code> except that they relate to pads instead of windows. The additional parameters are needed to indicate what part of the pad and screen are involved. <i>pminrow</i> and <i>pmincol</i> specify the upper left-hand corner of the rectangle to be displayed in the pad. <i>sminrow</i>, <i>smincol</i>, <i>smaxrow</i>, and <i>smaxcol</i> specify the edges of the rectangle to be displayed on the screen. The lower right-hand corner of the rectangle to be displayed in the pad is calculated from the screen coordinates, since the rectangles must be the same size. Both rectangles must be entirely contained within their respective structures. Negative values of <i>pminrow</i>, <i>pmincol</i>, <i>sminrow</i>, or <i>smincol</i> are treated as if they were zero.</p> |

prefresh(3CURSES)

The `pechochar()` routine is functionally equivalent to a call to `addch(3CURSES)` followed by a call to `refresh(3CURSES)`, a call to `waddch(3CURSES)` followed by a call to `wrefresh(3CURSES)`, or a call to `waddch(3CURSES)` followed by a call to `prefresh()`. The knowledge that only a single character is being output is taken into consideration and, for non-control characters, a considerable performance gain might be seen by using these routines instead of their equivalents. In the case of `pechochar()`, the last location of the pad on the screen is reused for the arguments to `prefresh()`.

RETURN VALUES Routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

Routines that return pointers return `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `addch(3CURSES)`, `curses(3CURSES)`, `refresh(3CURSES)`, `subwin(3CURSES)`, `touchline(3CURSES)`, `touchwin(3CURSES)`, `waddch(3CURSES)`, `wnoutrefresh(3CURSES)`, `wrefresh(3CURSES)`, `attributes(5)`

NOTES The header file `<curses.h>` automatically includes the header files `<stdio.h>`, `<unctrl.h>` and `<widec.h>`.

Note that `pechochar()` may be a macro.

prefresh(3XCURSES)

| | | | | | | | | | | | | | | | | | | | | | |
|--------------------|---|---------------|--|--------------|--|------------|-------------------------------------|----------------|--|----------------|--|----------------|---|----------------|--|----------------|--|----------------|---|-------------|---|
| NAME | newpad, pnoutrefresh, prefresh, subpad – create or refresh a pad or subpad | | | | | | | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> WINDOW *newpad(int nlines, int ncols); int pnoutrefresh(WINDOW *pad, int pminrow, int pmincol, int sminrow, int smincol, int smaxrow, int smaxcol); int prefresh(WINDOW *pad, int pminrow, int pmincol, int sminrow, int smincol, int smaxrow, int smaxcol); WINDOW *subpad(WINDOW *orig, int nlines, int ncols);</pre> | | | | | | | | | | | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>nlines</i></td><td>Is the number of lines in the pad to be created.</td></tr><tr><td><i>ncols</i></td><td>Is the number of columns in the pad to be created.</td></tr><tr><td><i>pad</i></td><td>Is a pointer to the pad to refresh.</td></tr><tr><td><i>pminrow</i></td><td>Is the row coordinate of the upper left corner of the pad rectangle to be copied</td></tr><tr><td><i>pmincol</i></td><td>Is the column coordinate of the upper left corner of the pad rectangle to be copied.</td></tr><tr><td><i>sminrow</i></td><td>Is the row coordinate of the upper left corner of the rectangle on the physical screen where pad is to be positioned.</td></tr><tr><td><i>smincol</i></td><td>Is the column coordinate of the upper left corner of the rectangle on the physical screen where pad is to be positioned.</td></tr><tr><td><i>smaxrow</i></td><td>Is the row coordinate of the lower right corner of the rectangle on the physical screen where the pad is to be positioned.</td></tr><tr><td><i>smaxcol</i></td><td>Is the column coordinate of the lower right corner of the rectangle on the physical screen where the pad is to be positioned.</td></tr><tr><td><i>orig</i></td><td>Is a pointer to the parent pad within which a sub-pad is created.</td></tr></table> | <i>nlines</i> | Is the number of lines in the pad to be created. | <i>ncols</i> | Is the number of columns in the pad to be created. | <i>pad</i> | Is a pointer to the pad to refresh. | <i>pminrow</i> | Is the row coordinate of the upper left corner of the pad rectangle to be copied | <i>pmincol</i> | Is the column coordinate of the upper left corner of the pad rectangle to be copied. | <i>sminrow</i> | Is the row coordinate of the upper left corner of the rectangle on the physical screen where pad is to be positioned. | <i>smincol</i> | Is the column coordinate of the upper left corner of the rectangle on the physical screen where pad is to be positioned. | <i>smaxrow</i> | Is the row coordinate of the lower right corner of the rectangle on the physical screen where the pad is to be positioned. | <i>smaxcol</i> | Is the column coordinate of the lower right corner of the rectangle on the physical screen where the pad is to be positioned. | <i>orig</i> | Is a pointer to the parent pad within which a sub-pad is created. |
| <i>nlines</i> | Is the number of lines in the pad to be created. | | | | | | | | | | | | | | | | | | | | |
| <i>ncols</i> | Is the number of columns in the pad to be created. | | | | | | | | | | | | | | | | | | | | |
| <i>pad</i> | Is a pointer to the pad to refresh. | | | | | | | | | | | | | | | | | | | | |
| <i>pminrow</i> | Is the row coordinate of the upper left corner of the pad rectangle to be copied | | | | | | | | | | | | | | | | | | | | |
| <i>pmincol</i> | Is the column coordinate of the upper left corner of the pad rectangle to be copied. | | | | | | | | | | | | | | | | | | | | |
| <i>sminrow</i> | Is the row coordinate of the upper left corner of the rectangle on the physical screen where pad is to be positioned. | | | | | | | | | | | | | | | | | | | | |
| <i>smincol</i> | Is the column coordinate of the upper left corner of the rectangle on the physical screen where pad is to be positioned. | | | | | | | | | | | | | | | | | | | | |
| <i>smaxrow</i> | Is the row coordinate of the lower right corner of the rectangle on the physical screen where the pad is to be positioned. | | | | | | | | | | | | | | | | | | | | |
| <i>smaxcol</i> | Is the column coordinate of the lower right corner of the rectangle on the physical screen where the pad is to be positioned. | | | | | | | | | | | | | | | | | | | | |
| <i>orig</i> | Is a pointer to the parent pad within which a sub-pad is created. | | | | | | | | | | | | | | | | | | | | |
| DESCRIPTION | The <code>newpad()</code> function creates a new pad with the specified number of lines and columns. A pointer to the new pad structure is returned. A pad differs from a window in that it is not restricted to the size of the physical screen. It is useful when only part of a large window will be displayed at any one time. | | | | | | | | | | | | | | | | | | | | |

prefresh(3XCURSES)

Automatic refreshes by scrolling or echoing of input do not take place when pads are used. Pads have their own refresh commands, `prefresh()` and `pnoutrefresh()`.

The `prefresh()` function copies the specified portion of the logical pad to the terminal screen. The parameters `pmincol` and `pminrow` specify the upper left corner of the rectangular area of the pad to be displayed. The lower right coordinate of the rectangular area of the pad that is to be displayed is calculated from the screen parameters (`sminrow`, `smincol`, `smaxrow`, `smaxcol`).

This function calls the `pnoutrefresh()` function to copy the specified portion of *pad* to the terminal screen and the `doupdate(3XCURSES)` function to do the actual update. The logical cursor is copied to the same location in the physical window unless `leaveok(3XCURSES)` is enabled (in which case, the cursor is placed in a position that the program finds convenient).

When outputting several pads at once, it is often more efficient to call the `pnoutrefresh()` and `doupdate()` functions directly. A call to `pnoutrefresh()` for each pad first, followed by only one call to `doupdate()` to update the screen, results in one burst of output, fewer characters sent, and less CPU time used.

The `subpad()` function creates a sub-pad within the pad *orig* with the specified number of lines and columns. A pointer to the new pad structure is returned. The sub-pad is positioned in the middle of *orig*. Any changes made to one pad affect the other. `touchwin(3XCURSES)` or `touchline(3XCURSES)` will likely have to be called on pad *orig* to correctly update the window.

RETURN VALUES

On success, the `newpad()` and `subpad()` functions returns a pointer to the new pad data structure. Otherwise, they return a null pointer.

On success, the `pnoutrefresh()` and `prefresh()` functions return `OK`. Otherwise, they return `ERR`.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO

`clearok(3XCURSES)`, `doupdate(3XCURSES)`, `is_linetouched(3XCURSES)`, `libcurses(3XCURSES)`, `pechochar(3XCURSES)`, `attributes(5)`, `standards(5)`

printw(3CURSES)

| NAME | <code> curs_printw, printw, wprintw, mvprintw, mvwprintw, vwprintw</code> – print formatted output in curses windows | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int printw(char *<i>fmt</i>, /* <i>arg</i> */ ...); int wprintw(WINDOW *<i>win</i>, char *<i>fmt</i>, /* <i>arg</i> */ ...); int mvprintw(int <i>y</i>, int <i>x</i>, char *<i>fmt</i>, /* <i>arg</i> */ ...); int mvwprintw(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>, char *<i>fmt</i>, /* <i>arg</i> */...); #include <varargs.h> int vwprintw(WINDOW *<i>win</i>, char *<i>fmt</i>, /* <i>varglist</i> */ ...);</pre> | | | | |
| DESCRIPTION | <p>The <code>printw()</code>, <code>wprintw()</code>, <code>mvprintw()</code>, and <code>mvwprintw()</code> routines are analogous to <code>printf()</code> (see <code>printf(3C)</code>). In effect, the string that would be output by <code>printf()</code> is output instead as though <code>waddstr()</code> were used on the given window.</p> <p>The <code>vwprintw()</code> routine is analogous to <code>vprintf()</code> (see <code>vprintf(3C)</code>) and performs a <code>wprintw()</code> using a variable argument list. The third argument is a <code>va_list</code>, a pointer to a list of arguments, as defined in <code><varargs.h></code>.</p> | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>printf(3C)</code> , <code>vprintf(3C)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . | | | | |

| NAME | mvprintw, mvwprintw, printw, wprintw – print formatted output window | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int mvprintw(int y, int x, char *fmt, ...); int mvwprintw(WINDOW *win, int y, int x, char *fmt, ...); int printw(char *fmt, ...); int wprintw(WINDOW *win, char *fmt, ...);</pre> | | | | | | |
| PARAMETERS | <p><i>y</i> Is the y (row) coordinate position of the string's placement in the window.</p> <p><i>x</i> Is the x (column) coordinate position of the string's placement in the window.</p> <p><i>fmt</i> Is a printf() format string.</p> <p><i>win</i> Is a pointer to the window in which the string is to be written.</p> | | | | | | |
| DESCRIPTION | The mvprintw(), mvwprintw(), printw(), and wprintw() functions are analogous to printf(3C). The effect of these functions is as though sprintf() were used to format the string, and then waddstr(3XCURSES) were used to add that multi-byte string to the current or specified window at the current or specified cursor position. | | | | | | |
| RETURN VALUES | Upon successful completion, these functions return OK. Otherwise, they return ERR. | | | | | | |
| ERRORS | No errors are defined. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | addnstr(3XCURSES), libcurses(3XCURSES), printf(3C), attributes(5), standards(5) | | | | | | |

putp(3CURSES)

| | |
|--------------------|--|
| NAME | <code> curs_terminfo, setupterm, setterm, set_curterm, del_curterm, restartterm, tparm, tputs, putp, vidputs, vidattr, mvcur, tigetflag, tigetnum, tigetstr</code> – curses interfaces to terminfo database |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> #include <term.h> int setupterm(char *term, int fildes, int *errret); int setterm(char *term); int set_curterm(TERMINAL *nterm); int del_curterm(TERMINAL *oterm); int restartterm(char *term, int fildes, int *errret); char *tparm(char *str, long int p1, long int p2, long int p3, long int p4, long int p5, long int p6, long int p7, long int p8, long int p9); int tputs(char *str, int affcnt, int (*putc)(char)); int putp(char *str); int vidputs(chtype attrs, int (*putc)(char)); int vidattr(chtype attrs); int mvcur(int oldrow, int oldcol, int newrow, int newcol); int tigetflag(char *capname); int tigetnum(char *capname); char *tigetstr(char *capname);</pre> |
| DESCRIPTION | <p>These low-level routines must be called by programs that have to deal directly with the <i>terminfo</i> database to handle certain terminal capabilities, such as programming function keys. For all other functionality, curses routines are more suitable and their use is recommended.</p> <p>Initially, <code>setupterm()</code> should be called. Note that <code>setupterm()</code> is automatically called by <code>initscr()</code> and <code>newterm()</code>. This defines the set of terminal-dependent variables (listed in <code>terminfo(4)</code>). The <i>terminfo</i> variables <code>lines</code> and <code>columns</code> are initialized by <code>setupterm()</code> as follows: If <code>use_env(FALSE)</code> has been called, values for <code>lines</code> and <code>columns</code> specified in <i>terminfo</i> are used. Otherwise, if the environment variables <code>LINES</code> and <code>COLUMNS</code> exist, their values are used. If these environment variables do not exist and the program is running in a window, the current window size is used. Otherwise, if the environment variables do not exist, the values for <code>lines</code> and <code>columns</code> specified in the <i>terminfo</i> database are used.</p> <p>The headers <code><curses.h></code> and <code><term.h></code> should be included (in this order) to get the definitions for these strings, numbers, and flags. Parameterized strings should be passed through <code>tparm()</code> to instantiate them. All <i>terminfo</i> strings (including the output</p> |

of `tparm()` should be printed with `tputs()` or `putp()`. Call the `reset_shell_mode()` routine to restore the tty modes before exiting (see `curs_kernel(3CURSES)`). Programs which use cursor addressing should output `enter_ca_mode` upon startup and should output `exit_ca_mode` before exiting. Programs desiring shell escapes should call `reset_shell_mode` and output `exit_ca_mode` before the shell is called and should output `enter_ca_mode` and call `reset_prog_mode` after returning from the shell.

The `setupterm()` routine reads in the *terminfo* database, initializing the *terminfo* structures, but does not set up the output virtualization structures used by *curses*. The terminal type is the character string *term*; if *term* is null, the environment variable `TERM` is used. All output is to file descriptor *fildev* which is initialized for output. If *errret* is not null, then `setupterm()` returns `OK` or `ERR` and stores a status value in the integer pointed to by *errret*. A status of 1 in *errret* is normal, 0 means that the terminal could not be found, and -1 means that the *terminfo* database could not be found. If *errret* is null, `setupterm()` prints an error message upon finding an error and exits. Thus, the simplest call is:

```
setupterm((char *)0, 1, (int *)0);
```

which uses all the defaults and sends the output to `stdout`.

The `setterm()` routine is being replaced by `setupterm()`. The call:

```
setupterm(term, 1, (int *)0)
```

provides the same functionality as `setterm(term)`. The `setterm()` routine is included here for compatibility and is supported at Level 2.

The `set_curterm()` routine sets the variable `cur_term` to *nterm*, and makes all of the *terminfo* boolean, numeric, and string variables use the values from *nterm*.

The `del_curterm()` routine frees the space pointed to by *oterm* and makes it available for further use. If *oterm* is the same as `cur_term`, references to any of the *terminfo* boolean, numeric, and string variables thereafter may refer to invalid memory locations until another `setupterm()` has been called.

The `restartterm()` routine is similar to `setupterm()` and `initscr()`, except that it is called after restoring memory to a previous state. It assumes that the windows and the input and output options are the same as when memory was saved, but the terminal type and baud rate may be different.

The `tparm()` routine instantiates the string *str* with parameters *pi*. A pointer is returned to the result of *str* with the parameters applied.

The `tputs()` routine applies padding information to the string *str* and outputs it. The *str* must be a *terminfo* string variable or the return value from `tparm()`, `tgetstr()`, or `tgoto()`. *affcnt* is the number of lines affected, or 1 if not applicable. *putc* is a `putchar()`-like routine to which the characters are passed, one at a time.

putp(3CURSES)

The `putp()` routine calls `tputs(str, 1, putchar)`. Note that the output of `putpA()` always goes to `stdout`, not to the *files* specified in `setupterm()`.

The `vidputs()` routine displays the string on the terminal in the video attribute mode *attrs*, which is any combination of the attributes listed in `curses(3CURSES)`. The characters are passed to the `putchar()`-like routine `putc()`.

The `vidattr()` routine is like the `vidputs()` routine, except that it outputs through `putchar()`.

The `mvcur()` routine provides low-level cursor motion.

The `tigetflag()`, `tigetnum()` and `tigetstr()` routines return the value of the capability corresponding to the *terminfo capname* passed to them, such as `xenl`.

With the `tigetflag()` routine, the value `-1` is returned if *capname* is not a boolean capability.

With the `tigetnum()` routine, the value `-2` is returned if *capname* is not a numeric capability.

With the `tigetstr()` routine, the value `(char *)-1` is returned if *capname* is not a string capability.

The *capname* for each capability is given in the table column entitled *capname code* in the capabilities section of `terminfo(4)`.

```
char *boolnames, *boolcodes, *boolfnames
char *numnames, *numcodes, *numfnames
char *strnames, *strcodes, *strfnames
```

These null-terminated arrays contain the *capnames*, the *termcap* codes, and the full C names, for each of the *terminfo* variables.

RETURN VALUES

All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

Routines that return pointers always return `NULL` on error.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_initscr(3CURSES)`, `curs_kernel(3CURSES)`, `curs_termcap(3CURSES)`, `curses(3CURSES)`, `putc(3C)`, `terminfo(4)`, `attributes(5)`

NOTES | The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

The `setupterm()` routine should be used in place of `setterm()`.

Note that `vidattr()` and `vidputs()` may be macros.

putp(3XCURSES)

| | |
|----------------------|--|
| NAME | putp, tputs – apply padding information and output string |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int putp(const char *str); int tputs(const char *str, int affcnt, int (*putfunc) (int));</pre> |
| PARAMETERS | <p><i>str</i> Is a pointer to a terminfo variable or return value from tgetstr(3XCURSES), tgoto(3XCURSES), tigetstr(3XCURSES), or tparm(3XCURSES).</p> <p><i>affcnt</i> Is the number of lines affected, or 1 if not relevant.</p> <p><i>putfunc</i> Is the output function.</p> |
| DESCRIPTION | <p>The putp() and tputs() functions are low-level functions used to deal directly with the terminfo database. The use of appropriate X/Open Curses functions is recommended for most situations.</p> <p>The tputs() function adds padding information and then outputs <i>str</i>. <i>str</i> must be a terminfo string variable or the result value from tgetstr(), tgoto(), tigetstr(), or tparm(). The tputs() function replaces the padding specification (if one exists) with enough characters to produce the specified delay. Characters are output one at a time to <i>putfunc</i>, a user-specified function similar to putchar(3C).</p> <p>The putp() function calls tputs() as follows:</p> <pre>tputs(str, 1, putchar)</pre> |
| RETURN VALUES | On success, these functions return OK. |
| ERRORS | None. |
| USAGE | The output of putp() goes to stdout, not to the file descriptor, <i>fdes</i> , specified in setupterm(3XCURSES). |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: |

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

putp(3XCURSES)

SEE ALSO | libcurses(3XCURSES), putchar(3C), setupterm(3XCURSES),
tgetent(3XCURSES), tigetflag(3XCURSES), terminfo(4), attributes(5),
standards(5)

putwin(3CURSES)

| | |
|--------------------|--|
| NAME | <code> curs_util, unctrl, keyname, filter, use_env, putwin, getwin, delay_output, flushingp </code> – curses miscellaneous utility routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> char *unctrl (chtype <i>c</i>); char *keyname (int <i>c</i>); int filter (void); void use_env (char <i>bool</i>); int putwin (WINDOW *<i>win</i>, FILE *<i>filep</i>); WINDOW *getwin (FILE *<i>filep</i>); int delay_output (int <i>ms</i>); int flushingp (void);</pre> |
| DESCRIPTION | <p>The <code>unctrl()</code> macro expands to a character string which is a printable representation of the character <code>c</code>. Control characters are displayed in the <code>^X</code> notation. Printing characters are displayed as is.</p> <p>With the <code>keyname()</code> routine, a character string corresponding to the key <code>c</code> is returned.</p> <p>The <code>filter()</code> routine, if used, is called before <code>initscr()</code> or <code>newterm()</code> are called. It makes <code>curses</code> think that there is a one-line screen. <code>curses</code> does not use any terminal capabilities that assume that they know on what line of the screen the cursor is positioned.</p> <p>The <code>use_env()</code> routine, if used, is called before <code>initscr()</code> or <code>newterm()</code> are called. When called with <code>FALSE</code> as an argument, the values of <code>lines</code> and <code>columns</code> specified in the <code>terminfo</code> database will be used, even if environment variables <code>LINES</code> and <code>COLUMNS</code> (used by default) are set, or if <code>curses</code> is running in a window (in which case default behavior would be to use the window size if <code>LINES</code> and <code>COLUMNS</code> are not set).</p> <p>With the <code>putwin()</code> routine, all data associated with window <code>win</code> is written into the file to which <code>filep</code> points. This information can be later retrieved using the <code>getwin()</code> function.</p> <p>The <code>getwin()</code> routine reads window related data stored in the file by <code>putwin()</code>. The routine then creates and initializes a new window using that data. It returns a pointer to the new window.</p> <p>The <code>delay_output()</code> routine inserts an <code>ms</code> millisecond pause in output. This routine should not be used extensively because padding characters are used rather than a CPU pause.</p> <p>The <code>flushinp()</code> routine throws away any typeahead that has been typed by the user and has not yet been read by the program.</p> |

putwin(3CURSES)

RETURN VALUES Except for `flushinp()`, routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`flushinp()` always returns `OK`.

Routines that return pointers return `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_initscr(3CURSES)`, `curl_scr_dump(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `unctrl()` is a macro, which is defined in `<unctrl.h>`.

putwin(3XCURSES)

NAME | `getwin`, `putwin` – read a window from, and write a window to, a file

SYNOPSIS |

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

WINDOW *getwin(FILE *filep);

int putwin(WINDOW *win, FILE *filep);
```

DESCRIPTION | The `getwin()` function reads window-related data (written earlier by `putwin()`) from the `stdio` stream pointed to by *filep*. It then creates and initializes a new window using that data.

The `putwin()` function writes all the data associated with the window pointed to by *win* to the `stdio` stream pointed to by *filep*. The `getwin()` function can later retrieve this data.

PARAMETERS | *filep* Is a pointer to a `stdio` stream.
win Is a pointer to a window.

RETURN VALUES | On success, the `getwin()` function returns a pointer to the new window created. Otherwise, it returns a null pointer.

On success, the `putwin()` function returns `OK`. Otherwise, it returns `ERR`.

ERRORS | None.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | `libcurses(3XCURSES)`, `scr_dump(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|---|
| NAME | curs_inopts, cbreak, nocbreak, echo, noecho, halfdelay, intrflush, keypad, meta, nodelay, notimeout, raw, noraw, noqiflush, qiflush, timeout, wtimeout, typeahead – curses terminal input option control routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int cbreak(void); int nocbreak(void); int echo(void); int noecho(void); int halfdelay(int tenths); int intrflush(WINDOW *win, bool bf); int keypad(WINDOW *win, bool bf); int meta(WINDOW *win, bool bf); int nodelay(WINDOW *win, bool bf); int notimeout(WINDOW *win, bool bf); int raw(void); int noraw(void); void noqiflush(void); void qiflush(void); void timeout(int delay); void wtimeout(WINDOW *win, int delay); int typeahead(int fildes);</pre> |
| DESCRIPTION | <p>The <code>cbreak()</code> and <code>nocbreak()</code> routines put the terminal into and out of <code>cbreak()</code> mode, respectively. In this mode, characters typed by the user are immediately available to the program, and erase/kill character-processing is not performed. When out of this mode, the tty driver buffers the typed characters until a newline or carriage return is typed. Interrupt and flow control characters are unaffected by this mode. Initially the terminal may or may not be in <code>cbreak()</code> mode, as the mode is inherited; therefore, a program should call <code>cbreak()</code> or <code>nocbreak()</code> explicitly. Most interactive programs using <code>curses</code> set the <code>cbreak()</code> mode.</p> <p>Note that <code>cbreak()</code> overrides <code>raw()</code>. (See <code>curs_getch(3CURSES)</code> for a discussion of how these routines interact with <code>echo()</code> and <code>noecho()</code>.)</p> <p>The <code>echo()</code> and <code>noecho()</code> routines control whether characters typed by the user are echoed by <code>getch()</code> as they are typed. Echoing by the tty driver is always disabled, but initially <code>getch()</code> is in <code>echo</code> mode, so characters typed are echoed. Authors of</p> |

qiflush(3CURSES)

most interactive programs prefer to do their own echoing in a controlled area of the screen, or not to echo at all, so they disable echoing by calling `noecho()`. (See `curs_getch(3CURSES)` for a discussion of how these routines interact with `cbreak()` and `nocbreak()`.)

The `halfdelay()` routine is used for half-delay mode, which is similar to `cbreak()` mode in that characters typed by the user are immediately available to the program. However, after blocking for *tenths* tenths of seconds, `ERR` is returned if nothing has been typed. The value of *tenths* must be a number between 1 and 255. Use `nocbreak()` to leave half-delay mode.

If the `intrflush()` option is enabled, (*bf* is `TRUE`), when an interrupt key is pressed on the keyboard (interrupt, break, quit) all output in the tty driver queue will be flushed, giving the effect of faster response to the interrupt, but causing `curses` to have the wrong idea of what is on the screen. Disabling (*bf* is `FALSE`), the option prevents the flush. The default for the option is inherited from the tty driver settings. The window argument is ignored.

The `keypad()` option enables the keypad of the user's terminal. If enabled (*bf* is `TRUE`), the user can press a function key (such as an arrow key) and `wgetch()` returns a single value representing the function key, as in `KEY_LEFT`. If disabled (*bf* is `FALSE`), `curses` does not treat function keys specially and the program has to interpret the escape sequences itself. If the keypad in the terminal can be turned on (made to transmit) and off (made to work locally), turning on this option causes the terminal keypad to be turned on when `wgetch()` is called. The default value for keypad is false.

Initially, whether the terminal returns 7 or 8 significant bits on input depends on the control mode of the tty driver (see `termio(7I)`). To force 8 bits to be returned, invoke `meta(win, TRUE)`. To force 7 bits to be returned, invoke `meta(win, FALSE)`. The window argument, *win*, is always ignored. If the terminfo capabilities `smm` (`meta_on`) and `rmm` (`meta_off`) are defined for the terminal, `smm` is sent to the terminal when `meta(win, TRUE)` is called and `rmm` is sent when `meta(win, FALSE)` is called.

The `nodelay()` option causes `getch()` to be a non-blocking call. If no input is ready, `getch()` returns `ERR`. If disabled (*bf* is `FALSE`), `getch()` waits until a key is pressed.

While interpreting an input escape sequence, `wgetch()` sets a timer while waiting for the next character. If `notimeout(win, TRUE)` is called, then `wgetch()` does not set a timer. The purpose of the timeout is to differentiate between sequences received from a function key and those typed by a user.

With the `raw()` and `noraw()` routines, the terminal is placed into or out of raw mode. Raw mode is similar to `cbreak()` mode, in that characters typed are immediately passed through to the user program. The differences are that in raw mode, the interrupt, quit, suspend, and flow control characters are all passed through uninterpreted, instead of generating a signal. The behavior of the `BREAK` key depends on other bits in the tty driver that are not set by `curses`.

qiflush(3CURSES)

When the `noqiflush()` routine is used, normal flush of input and output queues associated with the `INTR`, `QUIT` and `SUSP` characters will not be done (see `termio(7I)`). When `qiflush()` is called, the queues will be flushed when these control characters are read.

The `timeout()` and `wtimeout()` routines set blocking or non-blocking read for a given window. If *delay* is negative, blocking read is used (that is, waits indefinitely for input). If *delay* is zero, then non-blocking read is used (that is, read returns `ERR` if no input is waiting). If *delay* is positive, then read blocks for *delay* milliseconds, and returns `ERR` if there is still no input. Hence, these routines provide the same functionality as `nodelay()`, plus the additional capability of being able to block for only *delay* milliseconds (where *delay* is positive).

`curses` does “line-breakout optimization” by looking for typeahead periodically while updating the screen. If input is found, and it is coming from a `tty`, the current update is postponed until `refresh()` or `doupdate()` is called again. This allows faster response to commands typed in advance. Normally, the input `FILE` pointer passed to `newterm()`, or `stdin` in the case that `initscr()` was used, will be used to do this typeahead checking. The `typeahead()` routine specifies that the file descriptor *fildev* is to be used to check for typeahead instead. If *fildev* is `-1`, then no typeahead checking is done.

RETURN VALUES All routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_getch(3CURSES)`, `curs_initscr(3CURSES)`, `curses(3CURSES)`, `attributes(5)`, `termio(7I)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `echo()`, `noecho()`, `halfdelay()`, `intrflush()`, `meta()`, `nodelay()`, `notimeout()`, `noqiflush()`, `qiflush()`, `timeout()`, and `wtimeout()` may be macros.

qiflush(3XCURSES)

NAME noqiflush, qiflush – control flush of input and output on interrupt

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

void noqiflush(void);

void qiflush(void);
```

DESCRIPTION The `qiflush()` function enables the flushing of input and output queues when an interrupt, quit, or suspend character is sent to the terminal. The `noqiflush()` function disables this flushing.

RETURN VALUES These functions do not return a value.

ERRORS None

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `flushinp(3XCURSES)`, `intrflush(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | curs_inopts, cbreak, nocbreak, echo, noecho, halfdelay, intrflush, keypad, meta, nodelay, notimeout, raw, noraw, noqiflush, qiflush, timeout, wtimeout, typeahead – curses terminal input option control routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int cbreak(void); int nocbreak(void); int echo(void); int noecho(void); int halfdelay(int tenths); int intrflush(WINDOW *win, bool bf); int keypad(WINDOW *win, bool bf); int meta(WINDOW *win, bool bf); int nodelay(WINDOW *win, bool bf); int notimeout(WINDOW *win, bool bf); int raw(void); int noraw(void); void noqiflush(void); void qiflush(void); void timeout(int delay); void wtimeout(WINDOW *win, int delay); int typeahead(int fildes);</pre> |
| DESCRIPTION | <p>The <code>cbreak()</code> and <code>nocbreak()</code> routines put the terminal into and out of <code>cbreak()</code> mode, respectively. In this mode, characters typed by the user are immediately available to the program, and erase/kill character-processing is not performed. When out of this mode, the tty driver buffers the typed characters until a newline or carriage return is typed. Interrupt and flow control characters are unaffected by this mode. Initially the terminal may or may not be in <code>cbreak()</code> mode, as the mode is inherited; therefore, a program should call <code>cbreak()</code> or <code>nocbreak()</code> explicitly. Most interactive programs using <code>curses</code> set the <code>cbreak()</code> mode.</p> <p>Note that <code>cbreak()</code> overrides <code>raw()</code>. (See <code>curs_getch(3CURSES)</code> for a discussion of how these routines interact with <code>echo()</code> and <code>noecho()</code>.)</p> <p>The <code>echo()</code> and <code>noecho()</code> routines control whether characters typed by the user are echoed by <code>getch()</code> as they are typed. Echoing by the tty driver is always disabled, but initially <code>getch()</code> is in echo mode, so characters typed are echoed. Authors of</p> |

raw(3CURSES)

most interactive programs prefer to do their own echoing in a controlled area of the screen, or not to echo at all, so they disable echoing by calling `noecho()`. (See `curs_getch(3CURSES)` for a discussion of how these routines interact with `cbreak()` and `nocbreak()`.)

The `halfdelay()` routine is used for half-delay mode, which is similar to `cbreak()` mode in that characters typed by the user are immediately available to the program. However, after blocking for *tenths* tenths of seconds, `ERR` is returned if nothing has been typed. The value of *tenths* must be a number between 1 and 255. Use `nocbreak()` to leave half-delay mode.

If the `intrflush()` option is enabled, (*bf* is `TRUE`), when an interrupt key is pressed on the keyboard (interrupt, break, quit) all output in the tty driver queue will be flushed, giving the effect of faster response to the interrupt, but causing `curses` to have the wrong idea of what is on the screen. Disabling (*bf* is `FALSE`), the option prevents the flush. The default for the option is inherited from the tty driver settings. The window argument is ignored.

The `keypad()` option enables the keypad of the user's terminal. If enabled (*bf* is `TRUE`), the user can press a function key (such as an arrow key) and `wgetch()` returns a single value representing the function key, as in `KEY_LEFT`. If disabled (*bf* is `FALSE`), `curses` does not treat function keys specially and the program has to interpret the escape sequences itself. If the keypad in the terminal can be turned on (made to transmit) and off (made to work locally), turning on this option causes the terminal keypad to be turned on when `wgetch()` is called. The default value for keypad is false.

Initially, whether the terminal returns 7 or 8 significant bits on input depends on the control mode of the tty driver (see `termio(7I)`). To force 8 bits to be returned, invoke `meta(win, TRUE)`. To force 7 bits to be returned, invoke `meta(win, FALSE)`. The window argument, *win*, is always ignored. If the terminfo capabilities `smm` (`meta_on`) and `rmm` (`meta_off`) are defined for the terminal, `smm` is sent to the terminal when `meta(win, TRUE)` is called and `rmm` is sent when `meta(win, FALSE)` is called.

The `nodelay()` option causes `getch()` to be a non-blocking call. If no input is ready, `getch()` returns `ERR`. If disabled (*bf* is `FALSE`), `getch()` waits until a key is pressed.

While interpreting an input escape sequence, `wgetch()` sets a timer while waiting for the next character. If `notimeout(win, TRUE)` is called, then `wgetch()` does not set a timer. The purpose of the timeout is to differentiate between sequences received from a function key and those typed by a user.

With the `raw()` and `noraw()` routines, the terminal is placed into or out of raw mode. Raw mode is similar to `cbreak()` mode, in that characters typed are immediately passed through to the user program. The differences are that in raw mode, the interrupt, quit, suspend, and flow control characters are all passed through uninterpreted, instead of generating a signal. The behavior of the `BREAK` key depends on other bits in the tty driver that are not set by `curses`.

raw(3CURSES)

When the `noqiflush()` routine is used, normal flush of input and output queues associated with the `INTR`, `QUIT` and `SUSP` characters will not be done (see `termio(7I)`). When `qiflush()` is called, the queues will be flushed when these control characters are read.

The `timeout()` and `wtimeout()` routines set blocking or non-blocking read for a given window. If *delay* is negative, blocking read is used (that is, waits indefinitely for input). If *delay* is zero, then non-blocking read is used (that is, read returns `ERR` if no input is waiting). If *delay* is positive, then read blocks for *delay* milliseconds, and returns `ERR` if there is still no input. Hence, these routines provide the same functionality as `nodelay()`, plus the additional capability of being able to block for only *delay* milliseconds (where *delay* is positive).

`curses` does “line-breakout optimization” by looking for typeahead periodically while updating the screen. If input is found, and it is coming from a `tty`, the current update is postponed until `refresh()` or `doupdate()` is called again. This allows faster response to commands typed in advance. Normally, the input `FILE` pointer passed to `newterm()`, or `stdin` in the case that `initscr()` was used, will be used to do this typeahead checking. The `typeahead()` routine specifies that the file descriptor *fildev* is to be used to check for typeahead instead. If *fildev* is `-1`, then no typeahead checking is done.

RETURN VALUES

All routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_getch(3CURSES)`, `curs_initscr(3CURSES)`, `curses(3CURSES)`, `attributes(5)`, `termio(7I)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `echo()`, `noecho()`, `halfdelay()`, `intrflush()`, `meta()`, `nodelay()`, `notimeout()`, `noqiflush()`, `qiflush()`, `timeout()`, and `wtimeout()` may be macros.

raw(3XCURSES)

| NAME | <code>cbreak</code> , <code>nocbreak</code> , <code>noraw</code> , <code>raw</code> – set input mode controls | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int cbreak(void); int nocbreak(void); int noraw(void); int raw(void);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>cbreak()</code> function enables the character input mode. This overrides any previous call to the <code>raw()</code> function and turns the <code>stty</code> flag <code>ICANON</code> off.</p> <p>The <code>nocbreak()</code> function sets the line canonical mode and turns the <code>stty</code> flag <code>ICANON</code> on without touching the <code>ISIG</code> or <code>IXON</code> flags.</p> <p>The <code>noraw()</code> function sets the line canonical mode and turns the the <code>stty</code> flags <code>ICANON</code>, <code>ISIG</code>, and <code>IXON</code> all on.</p> <p>The <code>raw()</code> function sets the character input mode and turns the <code>stty</code> flags <code>ICANON</code>, <code>ISIG</code>, and <code>IXON</code> all off. This mode provides maximum control over input.</p> <p>It is important to remember that the terminal may or may not be in character mode operation initially. Most interactive programs require <code>cbreak()</code> to be enabled.</p> | | | | | | |
| RETURN VALUES | On success, these functions return <code>OK</code> . Otherwise, they return <code>ERR</code> . | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Standard</td></tr><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>getch(3XCURSES)</code> , <code>halfdelay(3XCURSES)</code> , <code>nodelay(3XCURSES)</code> , <code>timeout(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> , <code>termio(7I)</code> | | | | | | |

| | |
|--------------------|---|
| NAME | curs_refresh, refresh, wrefresh, wnoutrefresh, doupdate, redrawwin, wredrawln – refresh curses windows and lines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int refresh(void); int wrefresh(WINDOW *win); int wnoutrefresh(WINDOW *win); int doupdate(void); int redrawwin(WINDOW *win); int wredrawln(WINDOW *win, int beg_line, int num_lines);</pre> |
| DESCRIPTION | <p>The <code>refresh()</code> and <code>wrefresh()</code> routines (or <code>wnoutrefresh()</code> and <code>doupdate()</code>) must be called to get any output on the terminal, as other routines merely manipulate data structures. The routine <code>wrefresh()</code> copies the named window to the physical terminal screen, taking into account what is already there in order to do optimizations. The <code>refresh()</code> routine is the same, using <code>stdscr</code> as the default window. Unless <code>leaveok()</code> has been enabled, the physical cursor of the terminal is left at the location of the cursor for that window.</p> <p>The <code>wnoutrefresh()</code> and <code>doupdate()</code> routines allow multiple updates with more efficiency than <code>wrefresh()</code> alone. In addition to all the window structures, <code>curses</code> keeps two data structures representing the terminal screen: a physical screen, describing what is actually on the screen, and a virtual screen, describing what the programmer wants to have on the screen.</p> <p>The routine <code>wrefresh()</code> works by first calling <code>wnoutrefresh()</code>, which copies the named window to the virtual screen, and then calling <code>doupdate()</code>, which compares the virtual screen to the physical screen and does the actual update. If the programmer wishes to output several windows at once, a series of calls to <code>wrefresh()</code> results in alternating calls to <code>wnoutrefresh()</code> and <code>doupdate()</code>, causing several bursts of output to the screen. By first calling <code>wnoutrefresh()</code> for each window, it is then possible to call <code>doupdate()</code> once, resulting in only one burst of output, with fewer total characters transmitted and less CPU time used. If the <code>win</code> argument to <code>wrefresh()</code> is the global variable <code>curscr</code>, the screen is immediately cleared and repainted from scratch.</p> <p>The <code>redrawwin()</code> routine indicates to <code>curses</code> that some screen lines are corrupted and should be thrown away before anything is written over them. These routines could be used for programs such as editors, which want a command to redraw some part of the screen or the entire screen. The routine <code>wredrawln()</code> is preferred over <code>redrawwin()</code> where a noisy communication line exists and redrawing the entire window could be subject to even more communication noise. Just redrawing several lines offers the possibility that they would show up unblemished.</p> |

redrawwin(3CURSES)

RETURN VALUES All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_ouptops(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `refresh()` and `redrawwin()` may be macros.

| NAME | redrawwin, wredrawln – redraw screen or portion of screen | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int redrawwin(WINDOW *win); int wredrawln(WINDOW *win, int beg_line, int num_lines);</pre> | | | | | | |
| PARAMETERS | <p><i>win</i> Is a pointer to the window in which to redraw.</p> <p><i>beg_line</i> Is the first line to redraw.</p> <p><i>num_lines</i> Is the number of lines to redraw.</p> | | | | | | |
| DESCRIPTION | <p>The <code>redrawwin()</code> and <code>wredrawln()</code> functions force portions of a window to be redrawn to the terminal when the next refresh operation is performed.</p> <p>The <code>redrawwin()</code> function forces the entire window <i>win</i> to be redrawn, while the <code>wredrawln()</code> function forces only <i>num_lines</i> lines starting with <i>beg_line</i> to be redrawn. Normally, refresh operations use optimization methods to reduce the actual amount of the screen to redraw based on the current screen contents. These functions tell the refresh operations not to attempt any optimization when redrawing the indicated areas.</p> <p>These functions are useful when the data that exists on the screen is believed to be corrupt and for applications such as screen editors that redraw portions of the screen.</p> | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>doupdate(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

refresh(3CURSES)

| | |
|--------------------|---|
| NAME | <code> curs_refresh, refresh, wrefresh, wnoutrefresh, doupdate, redrawwin, wredrawln</code> – refresh curses windows and lines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int refresh(void); int wrefresh(WINDOW *win); int wnoutrefresh(WINDOW *win); int doupdate(void); int redrawwin(WINDOW *win); int wredrawln(WINDOW *win, int beg_line, int num_lines);</pre> |
| DESCRIPTION | <p>The <code>refresh()</code> and <code>wrefresh()</code> routines (or <code>wnoutrefresh()</code> and <code>doupdate()</code>) must be called to get any output on the terminal, as other routines merely manipulate data structures. The routine <code>wrefresh()</code> copies the named window to the physical terminal screen, taking into account what is already there in order to do optimizations. The <code>refresh()</code> routine is the same, using <code>stdscr</code> as the default window. Unless <code>leaveok()</code> has been enabled, the physical cursor of the terminal is left at the location of the cursor for that window.</p> <p>The <code>wnoutrefresh()</code> and <code>doupdate()</code> routines allow multiple updates with more efficiency than <code>wrefresh()</code> alone. In addition to all the window structures, <code>curses</code> keeps two data structures representing the terminal screen: a physical screen, describing what is actually on the screen, and a virtual screen, describing what the programmer wants to have on the screen.</p> <p>The routine <code>wrefresh()</code> works by first calling <code>wnoutrefresh()</code>, which copies the named window to the virtual screen, and then calling <code>doupdate()</code>, which compares the virtual screen to the physical screen and does the actual update. If the programmer wishes to output several windows at once, a series of calls to <code>wrefresh()</code> results in alternating calls to <code>wnoutrefresh()</code> and <code>doupdate()</code>, causing several bursts of output to the screen. By first calling <code>wnoutrefresh()</code> for each window, it is then possible to call <code>doupdate()</code> once, resulting in only one burst of output, with fewer total characters transmitted and less CPU time used. If the <code>win</code> argument to <code>wrefresh()</code> is the global variable <code>curscr</code>, the screen is immediately cleared and repainted from scratch.</p> <p>The <code>redrawwin()</code> routine indicates to <code>curses</code> that some screen lines are corrupted and should be thrown away before anything is written over them. These routines could be used for programs such as editors, which want a command to redraw some part of the screen or the entire screen. The routine <code>wredrawln()</code> is preferred over <code>redrawwin()</code> where a noisy communication line exists and redrawing the entire window could be subject to even more communication noise. Just redrawing several lines offers the possibility that they would show up unblemished.</p> |

refresh(3CURSES)

RETURN VALUES All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_ouptops(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `refresh()` and `redrawwin()` may be macros.

refresh(3XCURSES)

| NAME | douupdate, refresh, wnoutrefresh, wrefresh – refresh windows and lines | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int douupdate(void); int refresh(void); int wnoutrefresh(WINDOW *win); int wrefresh(WINDOW *win);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>refresh()</code> and <code>wrefresh()</code> functions copy <code>stdscr</code> and <code>win</code>, respectively, to the terminal screen. These functions call the <code>wnoutrefresh()</code> function to copy the specified window to <code>curscr</code> and the <code>douupdate()</code> function to do the actual update. The physical cursor is mapped to the same position as the logical cursor of the last window to update <code>curscr</code> unless <code>leaveok(3XCURSES)</code> is enabled (in which case, the cursor is placed in a position that X/Open Curses finds convenient).</p> <p>When outputting several windows at once, it is often more efficient to call the <code>wnoutrefresh()</code> and <code>douupdate()</code> functions directly. A call to <code>wnoutrefresh()</code> for each window, followed by only one call to <code>douupdate()</code> to update the screen, results in one burst of output, fewer characters sent, and less CPU time used.</p> <p>If the <code>win</code> parameter to <code>wrefresh()</code> is the global variable <code>curscr</code>, the screen is immediately cleared and repainted from scratch.</p> <p>For details on how the <code>wnoutrefresh()</code> function handles overlapping windows with broad glyphs, see the Overlapping Windows section of the <code>curses(3XCURSES)</code> reference manual page.</p> | | | | | | |
| PARAMETERS | <p><i>win</i> Is a pointer to the window in which to refresh.</p> | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Standard</td></tr><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>clearok(3XCURSES)</code> , <code>curses(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>prefresh(3XCURSES)</code> , <code>redrawwin(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

replace_panel(3CURSES)

NAME panel_window, replace_panel – get or set the current window of a panels panel

SYNOPSIS

```
cc [ flag ... ] file ... -lpanel -lcurses [ library .. ]
#include <panel.h>

WINDOW *panel_window(PANEL *panel);
int replace_panel(PANEL *panel, WINDOW *win);
```

DESCRIPTION panel_window() returns a pointer to the window of *panel*.
replace_panel() replaces the current window of *panel* with *win*.

RETURN VALUES panel_window() returns NULL on failure.
replace_panel() returns OK on successful completion, ERR otherwise.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), panels(3CURSES), attributes(5)

NOTES The header <panel.h> automatically includes the header <curses.h>.

reset_prog_mode(3CURSES)

| | |
|--------------------|--|
| NAME | <code> curs_kernel, def_prog_mode, def_shell_mode, reset_prog_mode, reset_shell_mode, resetty, savetty, getsyx, setsyx, ripoffline, curs_set, napms</code> – low-level curses routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int def_prog_mode(void); int def_shell_mode(void); int reset_prog_mode(void); int reset_shell_mode(void); int resetty(void); int savetty(void); int getsyx(int y, int x); int setsyx(int y, int x); int ripoffline(int line, int (*init)(WINDOW *, int)); int curs_set(int visibility); int napms(int ms);</pre> |
| DESCRIPTION | <p>The following routines give low-level access to various curses functionality. These routines typically are used inside library routines.</p> <p>The <code>def_prog_mode()</code> and <code>def_shell_mode()</code> routines save the current terminal modes as the “program” (in curses) or “shell” (not in curses) state for use by the <code>reset_prog_mode()</code> and <code>reset_shell_mode()</code> routines. This is done automatically by <code>initscr()</code>.</p> <p>The <code>reset_prog_mode()</code> and <code>reset_shell_mode()</code> routines restore the terminal to “program” (in curses) or “shell” (out of curses) state. These are done automatically by <code>endwin()</code> and, after an <code>endwin()</code>, by <code>doupdate()</code>, so they normally are not called.</p> <p>The <code>resetty()</code> and <code>savetty()</code> routines save and restore the state of the terminal modes. <code>savetty()</code> saves the current state in a buffer and <code>resetty()</code> restores the state to what it was at the last call to <code>savetty()</code>.</p> <p>With the <code>getsyx()</code> routine, the current coordinates of the virtual screen cursor are returned in <code>y</code> and <code>x</code>. If <code>leaveok()</code> is currently TRUE, then <code>-1,-1</code> is returned. If lines have been removed from the top of the screen, using <code>ripoffline()</code>, <code>y</code> and <code>x</code> include these lines; therefore, <code>y</code> and <code>x</code> should be used only as arguments for <code>setsyx()</code>.</p> <p>With the <code>setsyx()</code> routine, the virtual screen cursor is set to <code>y, x</code>. If <code>y</code> and <code>x</code> are both <code>-1</code>, then <code>leaveok()</code> is set. The two routines <code>getsyx()</code> and <code>setsyx()</code> are designed to be used by a library routine, which manipulates curses windows but does not</p> |

reset_prog_mode(3CURSES)

want to change the current position of the program's cursor. The library routine would call `getsyx()` at the beginning, do its manipulation of its own windows, do a `wnoutrefresh()` on its windows, call `setsyx()`, and then call `doupdate()`.

The `riporffline()` routine provides access to the same facility that `slk_init()` (see `curs_slk(3CURSES)`) uses to reduce the size of the screen. `riporffline()` must be called before `initscr()` or `newterm()` is called. If `line` is positive, a line is removed from the top of `stdscr()`; if `line` is negative, a line is removed from the bottom. When this is done inside `initscr()`, the routine `init()` (supplied by the user) is called with two arguments: a window pointer to the one-line window that has been allocated and an integer with the number of columns in the window. Inside this initialization routine, the integer variables `LINES` and `COLS` (defined in `<curses.h>`) are not guaranteed to be accurate and `wrefresh()` or `doupdate()` must not be called. It is allowable to call `wnoutrefresh()` during the initialization routine.

`riporffline()` can be called up to five times before calling `initscr()` or `newterm()`.

With the `curs_set()` routine, the cursor state is set to invisible, normal, or very visible for *visibility* equal to 0, 1, or 2 respectively. If the terminal supports the *visibility* requested, the previous *cursor* state is returned; otherwise, `ERR` is returned.

The `napms()` routine is used to sleep for *ms* milliseconds.

RETURN VALUES Except for `curs_set()`, these routines always return `OK`. `curs_set()` returns the previous cursor state, or `ERR` if the requested *visibility* is not supported.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_initscr(3CURSES)`, `curs_outopts(3CURSES)`, `curs_refresh(3CURSES)`, `curs_scr_dump(3CURSES)`, `curs_slk(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `getsyx()` is a macro, so an ampersand (&) is not necessary before the variables *y* and *x*.

reset_prog_mode(3XCURSES)

| NAME | def_prog_mode, def_shell_mode, reset_prog_mode, reset_shell_mode – save/restore terminal modes | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int def_prog_mode(void); int def_shell_mode(void); int reset_prog_mode(void); int reset_shell_mode(void);</pre> | | | | | | |
| DESCRIPTION | <p>The def_prog_mode() and def_shell_mode() functions save the current terminal modes as "program" (within X/Open Curses) or "shell" (outside X/Open Curses). The modes are saved automatically by initscr(3XCURSES), newterm(3XCURSES), and setupterm(3XCURSES).</p> <p>The reset_prog_mode() and reset_shell_mode() functions reset the current terminal modes to "program" (within X/Open Curses) or "shell" (outside X/Open Curses). The endwin(3XCURSES) function automatically calls the reset_shell_mode() function and the doupdate(3XCURSES) function calls the reset_prog_mode() function after calling endwin().</p> | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | endwin(3XCURSES), initscr(3XCURSES), libcurses(3XCURSES), newterm(3XCURSES), setupterm(3XCURSES), attributes(5), standards(5) | | | | | | |

| | |
|--------------------|--|
| NAME | curs_kernel, def_prog_mode, def_shell_mode, reset_prog_mode, reset_shell_mode, resetty, savetty, getsyx, setsyx, ripoffline, curs_set, napms – low-level curses routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int def_prog_mode(void); int def_shell_mode(void); int reset_prog_mode(void); int reset_shell_mode(void); int resetty(void); int savetty(void); int getsyx(int y, int x); int setsyx(int y, int x); int ripoffline(int line, int (*init)(WINDOW *, int)); int curs_set(int visibility); int napms(int ms);</pre> |
| DESCRIPTION | <p>The following routines give low-level access to various curses functionality. These routines typically are used inside library routines.</p> <p>The <code>def_prog_mode()</code> and <code>def_shell_mode()</code> routines save the current terminal modes as the “program” (in curses) or “shell” (not in curses) state for use by the <code>reset_prog_mode()</code> and <code>reset_shell_mode()</code> routines. This is done automatically by <code>initscr()</code>.</p> <p>The <code>reset_prog_mode()</code> and <code>reset_shell_mode()</code> routines restore the terminal to “program” (in curses) or “shell” (out of curses) state. These are done automatically by <code>endwin()</code> and, after an <code>endwin()</code>, by <code>doupdate()</code>, so they normally are not called.</p> <p>The <code>resetty()</code> and <code>savetty()</code> routines save and restore the state of the terminal modes. <code>savetty()</code> saves the current state in a buffer and <code>resetty()</code> restores the state to what it was at the last call to <code>savetty()</code>.</p> <p>With the <code>getsyx()</code> routine, the current coordinates of the virtual screen cursor are returned in <code>y</code> and <code>x</code>. If <code>leaveok()</code> is currently TRUE, then <code>-1,-1</code> is returned. If lines have been removed from the top of the screen, using <code>ripoffline()</code>, <code>y</code> and <code>x</code> include these lines; therefore, <code>y</code> and <code>x</code> should be used only as arguments for <code>setsyx()</code>.</p> <p>With the <code>setsyx()</code> routine, the virtual screen cursor is set to <code>y, x</code>. If <code>y</code> and <code>x</code> are both <code>-1</code>, then <code>leaveok()</code> is set. The two routines <code>getsyx()</code> and <code>setsyx()</code> are designed to be used by a library routine, which manipulates curses windows but does not</p> |

reset_shell_mode(3CURSES)

want to change the current position of the program's cursor. The library routine would call `getsyx()` at the beginning, do its manipulation of its own windows, do a `wnoutrefresh()` on its windows, call `setsyx()`, and then call `doupdate()`.

The `riporffline()` routine provides access to the same facility that `slk_init()` (see `curs_slk(3CURSES)`) uses to reduce the size of the screen. `riporffline()` must be called before `initscr()` or `newterm()` is called. If `line` is positive, a line is removed from the top of `stdscr()`; if `line` is negative, a line is removed from the bottom. When this is done inside `initscr()`, the routine `init()` (supplied by the user) is called with two arguments: a window pointer to the one-line window that has been allocated and an integer with the number of columns in the window. Inside this initialization routine, the integer variables `LINES` and `COLS` (defined in `<curses.h>`) are not guaranteed to be accurate and `wrefresh()` or `doupdate()` must not be called. It is allowable to call `wnoutrefresh()` during the initialization routine.

`riporffline()` can be called up to five times before calling `initscr()` or `newterm()`.

With the `curs_set()` routine, the cursor state is set to invisible, normal, or very visible for *visibility* equal to 0, 1, or 2 respectively. If the terminal supports the *visibility* requested, the previous *cursor* state is returned; otherwise, `ERR` is returned.

The `napms()` routine is used to sleep for *ms* milliseconds.

RETURN VALUES Except for `curs_set()`, these routines always return `OK`. `curs_set()` returns the previous cursor state, or `ERR` if the requested *visibility* is not supported.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_initscr(3CURSES)`, `curs_outopts(3CURSES)`, `curs_refresh(3CURSES)`, `curs_scr_dump(3CURSES)`, `curs_slk(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `getsyx()` is a macro, so an ampersand (&) is not necessary before the variables *y* and *x*.

NAME def_prog_mode, def_shell_mode, reset_prog_mode, reset_shell_mode – save/restore terminal modes

SYNOPSIS `cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \`
`-R /usr/xpg4/lib -lcurses [library...]`

`c89 [flag...] file... -lcurses [library...]`

`#include <curses.h>`

`int def_prog_mode(void);`

`int def_shell_mode(void);`

`int reset_prog_mode(void);`

`int reset_shell_mode(void);`

DESCRIPTION The `def_prog_mode()` and `def_shell_mode()` functions save the current terminal modes as "program" (within X/Open Curses) or "shell" (outside X/Open Curses). The modes are saved automatically by `initscr(3XCURSES)`, `newterm(3XCURSES)`, and `setupterm(3XCURSES)`.

The `reset_prog_mode()` and `reset_shell_mode()` functions reset the current terminal modes to "program" (within X/Open Curses) or "shell" (outside X/Open Curses). The `endwin(3XCURSES)` function automatically calls the `reset_shell_mode()` function and the `doupdate(3XCURSES)` function calls the `reset_prog_mode()` function after calling `endwin()`.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `endwin(3XCURSES)`, `initscr(3XCURSES)`, `libcurses(3XCURSES)`, `newterm(3XCURSES)`, `setupterm(3XCURSES)`, `attributes(5)`, `standards(5)`

resetty(3CURSES)

| | |
|--------------------|--|
| NAME | <code> curs_kernel, def_prog_mode, def_shell_mode, reset_prog_mode, reset_shell_mode, resetty, savetty, getsyx, setsyx, ripoffline, curs_set, napms</code> – low-level curses routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int def_prog_mode(void); int def_shell_mode(void); int reset_prog_mode(void); int reset_shell_mode(void); int resetty(void); int savetty(void); int getsyx(int y, int x); int setsyx(int y, int x); int ripoffline(int line, int (*init)(WINDOW *, int)); int curs_set(int visibility); int napms(int ms);</pre> |
| DESCRIPTION | <p>The following routines give low-level access to various curses functionality. These routines typically are used inside library routines.</p> <p>The <code>def_prog_mode()</code> and <code>def_shell_mode()</code> routines save the current terminal modes as the “program” (in curses) or “shell” (not in curses) state for use by the <code>reset_prog_mode()</code> and <code>reset_shell_mode()</code> routines. This is done automatically by <code>initscr()</code>.</p> <p>The <code>reset_prog_mode()</code> and <code>reset_shell_mode()</code> routines restore the terminal to “program” (in curses) or “shell” (out of curses) state. These are done automatically by <code>endwin()</code> and, after an <code>endwin()</code>, by <code>doupdate()</code>, so they normally are not called.</p> <p>The <code>resetty()</code> and <code>savetty()</code> routines save and restore the state of the terminal modes. <code>savetty()</code> saves the current state in a buffer and <code>resetty()</code> restores the state to what it was at the last call to <code>savetty()</code>.</p> <p>With the <code>getsyx()</code> routine, the current coordinates of the virtual screen cursor are returned in <code>y</code> and <code>x</code>. If <code>leaveok()</code> is currently TRUE, then <code>-1,-1</code> is returned. If lines have been removed from the top of the screen, using <code>ripoffline()</code>, <code>y</code> and <code>x</code> include these lines; therefore, <code>y</code> and <code>x</code> should be used only as arguments for <code>setsyx()</code>.</p> <p>With the <code>setsyx()</code> routine, the virtual screen cursor is set to <code>y, x</code>. If <code>y</code> and <code>x</code> are both <code>-1</code>, then <code>leaveok()</code> is set. The two routines <code>getsyx()</code> and <code>setsyx()</code> are designed to be used by a library routine, which manipulates curses windows but does not</p> |

resetty(3CURSES)

want to change the current position of the program's cursor. The library routine would call `getsyx()` at the beginning, do its manipulation of its own windows, do a `wnoutrefresh()` on its windows, call `setsyx()`, and then call `doupdate()`.

The `riporffline()` routine provides access to the same facility that `slk_init()` (see `curs_slk(3CURSES)`) uses to reduce the size of the screen. `riporffline()` must be called before `initscr()` or `newterm()` is called. If `line` is positive, a line is removed from the top of `stdscr()`; if `line` is negative, a line is removed from the bottom. When this is done inside `initscr()`, the routine `init()` (supplied by the user) is called with two arguments: a window pointer to the one-line window that has been allocated and an integer with the number of columns in the window. Inside this initialization routine, the integer variables `LINES` and `COLS` (defined in `<curses.h>`) are not guaranteed to be accurate and `wrefresh()` or `doupdate()` must not be called. It is allowable to call `wnoutrefresh()` during the initialization routine.

`riporffline()` can be called up to five times before calling `initscr()` or `newterm()`.

With the `curs_set()` routine, the cursor state is set to invisible, normal, or very visible for *visibility* equal to 0, 1, or 2 respectively. If the terminal supports the *visibility* requested, the previous *cursor* state is returned; otherwise, `ERR` is returned.

The `napms()` routine is used to sleep for *ms* milliseconds.

RETURN VALUES

Except for `curs_set()`, these routines always return `OK`. `curs_set()` returns the previous cursor state, or `ERR` if the requested *visibility* is not supported.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_initscr(3CURSES)`, `curs_outopts(3CURSES)`, `curs_refresh(3CURSES)`, `curs_scr_dump(3CURSES)`, `curs_slk(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `getsyx()` is a macro, so an ampersand (&) is not necessary before the variables *y* and *x*.

resetty(3XCURSES)

NAME resetty, savetty – restore/save terminal modes

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

int resetty(void);

int savetty(void);
```

DESCRIPTION The savetty() and resetty() functions save and restore the terminal state, respectively. The savetty() function saves the current state in a buffer; the resetty() function restores the state to that stored in the buffer at the time of the last savetty() call.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO libcurses(3XCURSES), attributes(5), standards(5)

restartterm(3CURSES)

| | |
|-------------|--|
| NAME | curs_terminfo, setupterm, setterm, set_curterm, del_curterm, restartterm, tparm, tputs, putp, vidputs, vidattr, mvcur, tigetflag, tigetnum, tigetstr – curses interfaces to terminfo database |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> #include <term.h> int setupterm(char *term, int fildes, int *errret); int setterm(char *term); int set_curterm(TERMINAL *nterm); int del_curterm(TERMINAL *oterm); int restartterm(char *term, int fildes, int *errret); char *tparm(char *str, long int p1, long int p2, long int p3, long int p4, long int p5, long int p6, long int p7, long int p8, long int p9); int tputs(char *str, int affcnt, int (*putc)(char)); int putp(char *str); int vidputs(chtype attrs, int (*putc)(char)); int vidattr(chtype attrs); int mvcur(int oldrow, int oldcol, int newrow, int newcol); int tigetflag(char *capname); int tigetnum(char *capname); char *tigetstr(char *capname);</pre> |
| DESCRIPTION | <p>These low-level routines must be called by programs that have to deal directly with the <i>terminfo</i> database to handle certain terminal capabilities, such as programming function keys. For all other functionality, curses routines are more suitable and their use is recommended.</p> <p>Initially, <code>setupterm()</code> should be called. Note that <code>setupterm()</code> is automatically called by <code>initscr()</code> and <code>newterm()</code>. This defines the set of terminal-dependent variables (listed in <code>terminfo(4)</code>). The <i>terminfo</i> variables <code>lines</code> and <code>columns</code> are initialized by <code>setupterm()</code> as follows: If <code>use_env(FALSE)</code> has been called, values for <code>lines</code> and <code>columns</code> specified in <i>terminfo</i> are used. Otherwise, if the environment variables <code>LINES</code> and <code>COLUMNS</code> exist, their values are used. If these environment variables do not exist and the program is running in a window, the current window size is used. Otherwise, if the environment variables do not exist, the values for <code>lines</code> and <code>columns</code> specified in the <i>terminfo</i> database are used.</p> <p>The headers <code><curses.h></code> and <code><term.h></code> should be included (in this order) to get the definitions for these strings, numbers, and flags. Parameterized strings should be passed through <code>tparm()</code> to instantiate them. All <i>terminfo</i> strings (including the output</p> |

restartterm(3CURSES)

of `tparm()` should be printed with `tputs()` or `putp()`. Call the `reset_shell_mode()` routine to restore the tty modes before exiting (see `curs_kernel(3CURSES)`). Programs which use cursor addressing should output `enter_ca_mode` upon startup and should output `exit_ca_mode` before exiting. Programs desiring shell escapes should call `reset_shell_mode` and output `exit_ca_mode` before the shell is called and should output `enter_ca_mode` and call `reset_prog_mode` after returning from the shell.

The `setupterm()` routine reads in the *terminfo* database, initializing the *terminfo* structures, but does not set up the output virtualization structures used by *curses*. The terminal type is the character string *term*; if *term* is null, the environment variable `TERM` is used. All output is to file descriptor *fildev* which is initialized for output. If *errret* is not null, then `setupterm()` returns `OK` or `ERR` and stores a status value in the integer pointed to by *errret*. A status of 1 in *errret* is normal, 0 means that the terminal could not be found, and -1 means that the *terminfo* database could not be found. If *errret* is null, `setupterm()` prints an error message upon finding an error and exits. Thus, the simplest call is:

```
setupterm((char *)0, 1, (int *)0);
```

which uses all the defaults and sends the output to `stdout`.

The `setterm()` routine is being replaced by `setupterm()`. The call:

```
setupterm(term, 1, (int *)0)
```

provides the same functionality as `setterm(term)`. The `setterm()` routine is included here for compatibility and is supported at Level 2.

The `set_curterm()` routine sets the variable `cur_term` to *nterm*, and makes all of the *terminfo* boolean, numeric, and string variables use the values from *nterm*.

The `del_curterm()` routine frees the space pointed to by *oterm* and makes it available for further use. If *oterm* is the same as `cur_term`, references to any of the *terminfo* boolean, numeric, and string variables thereafter may refer to invalid memory locations until another `setupterm()` has been called.

The `restartterm()` routine is similar to `setupterm()` and `initscr()`, except that it is called after restoring memory to a previous state. It assumes that the windows and the input and output options are the same as when memory was saved, but the terminal type and baud rate may be different.

The `tparm()` routine instantiates the string *str* with parameters *pi*. A pointer is returned to the result of *str* with the parameters applied.

The `tputs()` routine applies padding information to the string *str* and outputs it. The *str* must be a *terminfo* string variable or the return value from `tparm()`, `tgetstr()`, or `tgoto()`. *affcnt* is the number of lines affected, or 1 if not applicable. *putc* is a `putchar()`-like routine to which the characters are passed, one at a time.

restartterm(3CURSES)

The `putp()` routine calls `tputs(str, 1, putchar)`. Note that the output of `putpA()` always goes to `stdout`, not to the *files* specified in `setupterm()`.

The `vidputs()` routine displays the string on the terminal in the video attribute mode *attrs*, which is any combination of the attributes listed in `curses(3CURSES)`. The characters are passed to the `putchar()`-like routine `putc()`.

The `vidattr()` routine is like the `vidputs()` routine, except that it outputs through `putchar()`.

The `mvcur()` routine provides low-level cursor motion.

The `tigetflag()`, `tigetnum()` and `tigetstr()` routines return the value of the capability corresponding to the *terminfo capname* passed to them, such as `xenl`.

With the `tigetflag()` routine, the value `-1` is returned if *capname* is not a boolean capability.

With the `tigetnum()` routine, the value `-2` is returned if *capname* is not a numeric capability.

With the `tigetstr()` routine, the value `(char *)-1` is returned if *capname* is not a string capability.

The *capname* for each capability is given in the table column entitled *capname* code in the capabilities section of `terminfo(4)`.

```
char *boolnames, *boolcodes, *boolfnames
char *numnames, *numcodes, *numfnames
char *strnames, *strcodes, *strfnames
```

These null-terminated arrays contain the *capnames*, the *termcap* codes, and the full C names, for each of the *terminfo* variables.

RETURN VALUES

All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

Routines that return pointers always return `NULL` on error.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_initscr(3CURSES)`, `curs_kernel(3CURSES)`, `curs_termcap(3CURSES)`, `curses(3CURSES)`, `putc(3C)`, `terminfo(4)`, `attributes(5)`

restartterm(3CURSES)

NOTES | The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

The `setupterm()` routine should be used in place of `setterm()`.

Note that `vidattr()` and `vidputs()` may be macros.

| NAME | del_curterm, restartterm, set_curterm, setupterm – interfaces to the terminfo database | | | | | | | | |
|--------------------|--|-------|-------------|---|--------|---|-----------------------------|----|---|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <term.h> int del_curterm(TERMINAL *oterm); int restartterm(char *term, int fildes, int *errret); TERMINAL *set_curterm(TERMINAL *nterm); int setupterm(char *term, int fildes, int *errret);</pre> | | | | | | | | |
| DESCRIPTION | <p>Within X/Open Curses, the <code>setupterm()</code> function is automatically called by the <code>initscr(3XC)</code> and <code>newterm(3XC)</code> functions. This function can be also be used outside of X/Open Curses when a program has to deal directly with the <code>terminfo</code> database to handle certain terminal capabilities. The use of appropriate X/Open Curses functions is recommended in all other situations.</p> <p>The <code>setupterm()</code> function loads terminal-dependent variables for the <code>terminfo</code> layer of X/Open Curses. The <code>setupterm()</code> function initializes the <code>terminfo</code> variables <code>lines</code> and <code>columns</code> such that if <code>use_env(FALSE)</code> has been called, the <code>terminfo</code> values assigned in the database are used regardless of the environmental variables <code>LINES</code> and <code>COLUMNS</code> or the program's window dimensions; when <code>use_env(TRUE)</code> has been called, which is the default, the environment variables <code>LINES</code> and <code>COLUMNS</code> are used, if they exist. If the environment variables do not exist and the program is running in a window, the current window size is used.</p> <p>The <code>term</code> parameter of <code>setupterm()</code> specifies the terminal; if null, terminal type is taken from the <code>TERM</code> environment variable. All output is sent to <code>fildes</code> which is initialized for output. If <code>errret</code> is not null, <code>OK</code> or <code>ERR</code> is returned and a status value is stored in the integer pointed to by <code>errret</code>. The following status values may be returned:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Value</th> <th style="text-align: left;">Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Normal</td> </tr> <tr> <td>0</td> <td>Terminal could not be found</td> </tr> <tr> <td>-1</td> <td><code>terminfo</code> database could not be found</td> </tr> </tbody> </table> <p>If <code>errret</code> is null, an error message is printed, and the <code>setupterm()</code> function calls the <code>exit()</code> function with a non-zero parameter.</p> | Value | Description | 1 | Normal | 0 | Terminal could not be found | -1 | <code>terminfo</code> database could not be found |
| Value | Description | | | | | | | | |
| 1 | Normal | | | | | | | | |
| 0 | Terminal could not be found | | | | | | | | |
| -1 | <code>terminfo</code> database could not be found | | | | | | | | |

restartterm(3XCURSES)

The `set_curterm()` function sets the `cur_term` variable to `nterm`. The values from `nterm` as well as other state information for the terminal are used by X/Open Curses functions such as `beep(3XCURSES)`, `flash(3XCURSES)`, `mvcur(3XCURSES)`, `tigetflag(3XCURSES)`, `tigetstr(3XCURSES)`, and `tigetnum(3XCURSES)`.

The `del_curterm()` function frees the space pointed to by `oterm`. If `oterm` and the `cur_term` variable are the same, all Boolean, numeric, or string terminfo variables will refer to invalid memory locations until you call `setupterm()` and specify a new terminal type.

The `restartterm()` function assumes that a call to `setupterm()` has already been made (probably from `initscr()` or `newterm()`). It allows you to specify a new terminal type in `term` and updates the data returned by `baudrate(3XCURSES)` based on `fildev`. Other information created by the `initscr()`, `newterm()`, and `setupterm()` functions is preserved.

| | | |
|-------------------|---------------|---|
| PARAMETERS | <i>oterm</i> | Is the terminal type for which to free space. |
| | <i>term</i> | Is the terminal type for which variables are set. |
| | <i>fildev</i> | Is a file descriptor initialized for output. |
| | <i>errret</i> | Is a pointer to an integer in which the status value is stored. |
| | <i>nterm</i> | Is the new terminal to become the current terminal. |

RETURN VALUES On success, the `set_curterm()` function returns the previous value of `cur_term`. Otherwise, it returns a null pointer.

On success, the other functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `baudrate(3XCURSES)`, `beep(3XCURSES)`, `initscr(3XCURSES)`, `libcurses(3XCURSES)`, `mvcur(3XCURSES)`, `tigetflag(3XCURSES)`, `use_env(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | curs_kernel, def_prog_mode, def_shell_mode, reset_prog_mode, reset_shell_mode, resetty, savetty, getsyx, setsyx, ripoffline, curs_set, napms – low-level curses routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int def_prog_mode(void); int def_shell_mode(void); int reset_prog_mode(void); int reset_shell_mode(void); int resetty(void); int savetty(void); int getsyx(int y, int x); int setsyx(int y, int x); int ripoffline(int line, int (*init)(WINDOW *, int)); int curs_set(int visibility); int napms(int ms);</pre> |
| DESCRIPTION | <p>The following routines give low-level access to various curses functionality. These routines typically are used inside library routines.</p> <p>The <code>def_prog_mode()</code> and <code>def_shell_mode()</code> routines save the current terminal modes as the “program” (in curses) or “shell” (not in curses) state for use by the <code>reset_prog_mode()</code> and <code>reset_shell_mode()</code> routines. This is done automatically by <code>initscr()</code>.</p> <p>The <code>reset_prog_mode()</code> and <code>reset_shell_mode()</code> routines restore the terminal to “program” (in curses) or “shell” (out of curses) state. These are done automatically by <code>endwin()</code> and, after an <code>endwin()</code>, by <code>doupdate()</code>, so they normally are not called.</p> <p>The <code>resetty()</code> and <code>savetty()</code> routines save and restore the state of the terminal modes. <code>savetty()</code> saves the current state in a buffer and <code>resetty()</code> restores the state to what it was at the last call to <code>savetty()</code>.</p> <p>With the <code>getsyx()</code> routine, the current coordinates of the virtual screen cursor are returned in <code>y</code> and <code>x</code>. If <code>leaveok()</code> is currently TRUE, then <code>-1,-1</code> is returned. If lines have been removed from the top of the screen, using <code>ripoffline()</code>, <code>y</code> and <code>x</code> include these lines; therefore, <code>y</code> and <code>x</code> should be used only as arguments for <code>setsyx()</code>.</p> <p>With the <code>setsyx()</code> routine, the virtual screen cursor is set to <code>y, x</code>. If <code>y</code> and <code>x</code> are both <code>-1</code>, then <code>leaveok()</code> is set. The two routines <code>getsyx()</code> and <code>setsyx()</code> are designed to be used by a library routine, which manipulates curses windows but does not</p> |

ripoffline(3CURSES)

want to change the current position of the program's cursor. The library routine would call `getsyx()` at the beginning, do its manipulation of its own windows, do a `wnoutrefresh()` on its windows, call `setsyx()`, and then call `doupdate()`.

The `ripoffline()` routine provides access to the same facility that `slk_init()` (see `curs_slk(3CURSES)`) uses to reduce the size of the screen. `ripoffline()` must be called before `initscr()` or `newterm()` is called. If `line` is positive, a line is removed from the top of `stdscr()`; if `line` is negative, a line is removed from the bottom. When this is done inside `initscr()`, the routine `init()` (supplied by the user) is called with two arguments: a window pointer to the one-line window that has been allocated and an integer with the number of columns in the window. Inside this initialization routine, the integer variables `LINES` and `COLS` (defined in `<curses.h>`) are not guaranteed to be accurate and `wrefresh()` or `doupdate()` must not be called. It is allowable to call `wnoutrefresh()` during the initialization routine.

`ripoffline()` can be called up to five times before calling `initscr()` or `newterm()`.

With the `curs_set()` routine, the cursor state is set to invisible, normal, or very visible for *visibility* equal to 0, 1, or 2 respectively. If the terminal supports the *visibility* requested, the previous *cursor* state is returned; otherwise, `ERR` is returned.

The `napms()` routine is used to sleep for *ms* milliseconds.

RETURN VALUES

Except for `curs_set()`, these routines always return `OK`. `curs_set()` returns the previous cursor state, or `ERR` if the requested *visibility* is not supported.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_initscr(3CURSES)`, `curs_outopts(3CURSES)`, `curs_refresh(3CURSES)`, `curs_scr_dump(3CURSES)`, `curs_slk(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `getsyx()` is a macro, so an ampersand (&) is not necessary before the variables *y* and *x*.

NAME | riproffline – reserve screen line for dedicated purpose

SYNOPSIS | `cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \`
`-R /usr/xpg4/lib -lcurses [library...]`

PARAMETERS | `c89 [flag...] file... -lcurses [library...]`
`#include <curses.h>`
`int riproffline(int line, int (*init)(WINDOW *win, int width));`

PARAMETERS | *line* | determines whether the screen line being reserved comes from the top of `stdscr` (line is positive) or the bottom (line is negative).

init | Is a pointer to a function that initializes the one-line window.

win | Is a pointer to one-line window created by this function.

width | Is the number of columns in the window pointed to by the *win* parameter.

DESCRIPTION | The `riproffline()` function reserves a screen line as a one line window.

To use this function, it must be called before you call `initscr(3XCURSES)` or `newterm(3XCURSES)`. When `initscr()` or `newterm()` is called, so is the function pointed to by `init`. The function pointed to by `init` takes two arguments: a pointer to the one-line window and the number of columns in that window. This function cannot use the `LINES` or `COLS` variables and cannot call `wrefresh(3XCURSES)` or `doupdate(3XCURSES)`, but may call `wnoutrefresh(3XCURSES)`.

RETURN VALUES | The `riproffline()` function always returns OK.

ERRORS | None.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | `doupdate(3XCURSES)`, `initscr(3XCURSES)`, `libcurses(3XCURSES)`, `slk_attroff(3XCURSES)`, `attributes(5)`, `standards(5)`

savetty(3CURSES)

| | |
|--------------------|--|
| NAME | <code> curs_kernel, def_prog_mode, def_shell_mode, reset_prog_mode, reset_shell_mode, resetty, savetty, getsyx, setsyx, ripoffline, curs_set, napms</code> – low-level curses routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int def_prog_mode(void); int def_shell_mode(void); int reset_prog_mode(void); int reset_shell_mode(void); int resetty(void); int savetty(void); int getsyx(int y, int x); int setsyx(int y, int x); int ripoffline(int line, int (*init)(WINDOW *, int)); int curs_set(int visibility); int napms(int ms);</pre> |
| DESCRIPTION | <p>The following routines give low-level access to various curses functionality. These routines typically are used inside library routines.</p> <p>The <code>def_prog_mode()</code> and <code>def_shell_mode()</code> routines save the current terminal modes as the “program” (in curses) or “shell” (not in curses) state for use by the <code>reset_prog_mode()</code> and <code>reset_shell_mode()</code> routines. This is done automatically by <code>initscr()</code>.</p> <p>The <code>reset_prog_mode()</code> and <code>reset_shell_mode()</code> routines restore the terminal to “program” (in curses) or “shell” (out of curses) state. These are done automatically by <code>endwin()</code> and, after an <code>endwin()</code>, by <code>doupdate()</code>, so they normally are not called.</p> <p>The <code>resetty()</code> and <code>savetty()</code> routines save and restore the state of the terminal modes. <code>savetty()</code> saves the current state in a buffer and <code>resetty()</code> restores the state to what it was at the last call to <code>savetty()</code>.</p> <p>With the <code>getsyx()</code> routine, the current coordinates of the virtual screen cursor are returned in <code>y</code> and <code>x</code>. If <code>leaveok()</code> is currently TRUE, then <code>-1,-1</code> is returned. If lines have been removed from the top of the screen, using <code>ripoffline()</code>, <code>y</code> and <code>x</code> include these lines; therefore, <code>y</code> and <code>x</code> should be used only as arguments for <code>setsyx()</code>.</p> <p>With the <code>setsyx()</code> routine, the virtual screen cursor is set to <code>y, x</code>. If <code>y</code> and <code>x</code> are both <code>-1</code>, then <code>leaveok()</code> is set. The two routines <code>getsyx()</code> and <code>setsyx()</code> are designed to be used by a library routine, which manipulates curses windows but does not</p> |

savetty(3CURSES)

want to change the current position of the program's cursor. The library routine would call `getsyx()` at the beginning, do its manipulation of its own windows, do a `wnoutrefresh()` on its windows, call `setsyx()`, and then call `doupdate()`.

The `riporffline()` routine provides access to the same facility that `slk_init()` (see `curs_slk(3CURSES)`) uses to reduce the size of the screen. `riporffline()` must be called before `initscr()` or `newterm()` is called. If `line` is positive, a line is removed from the top of `stdscr()`; if `line` is negative, a line is removed from the bottom. When this is done inside `initscr()`, the routine `init()` (supplied by the user) is called with two arguments: a window pointer to the one-line window that has been allocated and an integer with the number of columns in the window. Inside this initialization routine, the integer variables `LINES` and `COLS` (defined in `<curses.h>`) are not guaranteed to be accurate and `wrefresh()` or `doupdate()` must not be called. It is allowable to call `wnoutrefresh()` during the initialization routine.

`riporffline()` can be called up to five times before calling `initscr()` or `newterm()`.

With the `curs_set()` routine, the cursor state is set to invisible, normal, or very visible for *visibility* equal to 0, 1, or 2 respectively. If the terminal supports the *visibility* requested, the previous *cursor* state is returned; otherwise, `ERR` is returned.

The `napms()` routine is used to sleep for *ms* milliseconds.

RETURN VALUES Except for `curs_set()`, these routines always return `OK`. `curs_set()` returns the previous cursor state, or `ERR` if the requested *visibility* is not supported.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_initscr(3CURSES)`, `curs_outopts(3CURSES)`, `curs_refresh(3CURSES)`, `curs_scr_dump(3CURSES)`, `curs_slk(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `getsyx()` is a macro, so an ampersand (&) is not necessary before the variables *y* and *x*.

savetty(3XCURSES)

NAME | resetty, savetty – restore/save terminal modes

SYNOPSIS | `cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \`
| `-R /usr/xpg4/lib -lcurses [library...]`

| `c89 [flag...] file... -lcurses [library...]`

| `#include <curses.h>`
 | `int resetty(void);`
 | `int savetty(void);`

DESCRIPTION | The `savetty()` and `resetty()` functions save and restore the terminal state, respectively. The `savetty()` function saves the current state in a buffer; the `resetty()` function restores the state to that stored in the buffer at the time of the last `savetty()` call.

RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR.

ERRORS | None.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

scale_form(3CURSES)

NAME form_win, set_form_win, set_form_sub, form_sub, scale_form – forms window and subwindow association routines

SYNOPSIS

```
cc [ flag... ] file... -lform -lcurses [ library... ]
#include <form.h>

int set_form_win(FORM *form, WINDOW *win);
WINDOW *form_win(FORM *form);
int set_form_sub(FORM *form, WINDOW *sub);
WINDOW *form_sub(FORM *form);
int scale_form(FORM *form, int *rows, int *cols);
```

DESCRIPTION set_form_win() sets the window of *form* to *win*. form_win() returns a pointer to the window associated with *form*. set_form_sub() sets the subwindow of *form* to *sub*. form_sub() returns a pointer to the subwindow associated with *form*. scale_form() returns the smallest window size necessary for the subwindow of *form*. *rows* and *cols* are pointers to the locations used to return the number of rows and columns for the form.

RETURN VALUES Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:

| | |
|-----------------|---------------------------------------|
| E_OK | The function returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An argument is incorrect. |
| E_NOT_CONNECTED | The field is not connected to a form. |
| E_POSTED | The form is posted. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), forms(3CURSES), attributes(5)

NOTES The header <form.h> automatically includes the headers <eti.h> and <curses.h>.

scale_menu(3CURSES)

| NAME | menu_win, set_menu_win, set_menu_sub, menu_sub, scale_menu – menus window and subwindow association routines | | | | | | | | | | |
|----------------------|---|----------------|------------------------------------|----------------|---------------|----------------|--|----------|-----------------------------|-----------------|-------------------------------------|
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int set_menu_win(MENU *menu, WINDOW *win); WINDOW *menu_win(MENU *menu); int set_menu_sub(MENU *menu, WINDOW *sub); WINDOW *menu_sub(MENU *menu); int scale_window(MENU *menu, int *rows, int *cols);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>set_menu_win() sets the window of <i>menu</i> to <i>win</i>. menu_win() returns a pointer to the window of <i>menu</i>. set_menu_sub() sets the subwindow of <i>menu</i> to <i>sub</i>. menu_sub() returns a pointer to the subwindow of <i>menu</i>. scale_window() returns the minimum window size necessary for the subwindow of <i>menu</i>. <i>rows</i> and <i>cols</i> are pointers to the locations used to return the values.</p> | | | | | | | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <table><tr><td>E_OK</td><td>The routine returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr><tr><td>E_BAD_ARGUMENT</td><td>An incorrect argument was passed to the routine.</td></tr><tr><td>E_POSTED</td><td>The menu is already posted.</td></tr><tr><td>E_NOT_CONNECTED</td><td>No items are connected to the menu.</td></tr></table> | E_OK | The routine returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An incorrect argument was passed to the routine. | E_POSTED | The menu is already posted. | E_NOT_CONNECTED | No items are connected to the menu. |
| E_OK | The routine returned successfully. | | | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | | | |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. | | | | | | | | | | |
| E_POSTED | The menu is already posted. | | | | | | | | | | |
| E_NOT_CONNECTED | No items are connected to the menu. | | | | | | | | | | |
| ATTRIBUTES | <p>See attributes(5) for descriptions of the following attributes:</p> <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | | | | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | | | | | |
| MT-Level | Unsafe | | | | | | | | | | |
| SEE ALSO | curses(3CURSES), menus(3CURSES), attributes(5) | | | | | | | | | | |
| NOTES | The header <menu.h> automatically includes the headers <eti.h> and <curses.h>. | | | | | | | | | | |

| NAME | curs_scanw, scanw, wscanw, mvscanw, mvwscanw, vwscanw – convert formatted input from a curses widow | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int scanw(char *fmt, /* arg */ ...); int wscanw(WINDOW *win, char *fmt, /* arg */ ...); int mvscanw(int y, int x, char *fmt, /* arg */ ...); int mvwscanw(WINDOW *win, int y, int x, char *fmt, /* arg */...); int vwscanw(WINDOW *win, char *fmt, va_list varlist);</pre> | | | | |
| DESCRIPTION | <p>The scanw(), wscanw(), and mvscanw() routines correspond to scanf() (see scanf(3C)). The effect of these routines is as though wgetstr() were called on the window, and the resulting line used as input for the scan. Fields which do not map to a variable in the fmt field are lost.</p> <p>The vwscanw() routine is similar to vwprintw() in that it performs a wscanw() using a variable argument list. The third argument is a va_list, a pointer to a list of arguments, as defined in <varargs.h>.</p> | | | | |
| RETURN VALUES | <p>vwscanw() returns ERR on failure and an integer equal to the number of fields scanned on success.</p> <p>Applications may interrogate the return value from the scanw, wscanw(), mvscanw(), and mvwscanw() routines to determine the number of fields which were mapped in the call.</p> | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curs_getstr(3CURSES), curs_printw(3CURSES), curses(3CURSES), scanf(3C), attributes(5) | | | | |
| NOTES | The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>. | | | | |

scanw(3XCURSES)

NAME mvscanw, mvwscanw, scanw, wscanw – convert formatted input from a window

SYNOPSIS `cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \`
`-R /usr/xpg4/lib -lcurses [library...]`

c89 `[flag...] file... -lcurses [library...]`

```
#include <curses.h>
int mvscanw(int y, int x, char *fmt, ...);
int mvwscanw(WINDOW *win, int y, int x, char *fmt, ...);
int scanw(char *fmt, ...);
int wscanw(WINDOW *win, char *fmt, ...);
```

PARAMETERS

y Is the y (row) coordinate of the position of the character to be read.

x Is the x (column) coordinate of the position of the character to be read.

fmt Is a scanf () format string.

win Is a pointer to the window in which the character is to be read.

DESCRIPTION These functions are similar to scanf(3C). Their effect is as though mvwgetstr(3XCURSES) were called to get a multi-byte character string from the current or specified window at the current or specified cursor position, and then sscanf () were used to interpret and convert that string.

RETURN VALUES Upon successful completion, these functions return OK. Otherwise, they return ERR.

ERRORS No errors are defined.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO getnstr(3XCURSES), libcurses(3XCURSES), printw(3XCURSES), scanf(3C), wcstombs(3C), attributes(5), standards(5)

| NAME | curs_scr_dump, scr_dump, scr_restore, scr_init, scr_set – read (write) a curses screen from (to) a file | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int scr_dump(char *filename); int scr_restore(char *filename); int scr_init(char *filename); int scr_set(char *filename);</pre> | | | | |
| DESCRIPTION | <p>With the <code>scr_dump()</code> routine, the current contents of the virtual screen are written to the file <i>filename</i>.</p> <p>With the <code>scr_restore()</code> routine, the virtual screen is set to the contents of <i>filename</i>, which must have been written using <code>scr_dump()</code>. The next call to <code>doupdate()</code> restores the screen to the way it looked in the dump file.</p> <p>With the <code>scr_init()</code> routine, the contents of <i>filename</i> are read in and used to initialize the curses data structures about what the terminal currently has on its screen. If the data is determined to be valid, curses bases its next update of the screen on this information rather than clearing the screen and starting from scratch. <code>scr_init()</code> is used after <code>initscr()</code> or a <code>system(3C)</code> call to share the screen with another process which has done a <code>scr_dump()</code> after its <code>endwin()</code> call. The data is declared invalid if the time-stamp of the tty is old or the terminfo capabilities <code>rmcup()</code> and <code>nrrmc()</code> exist.</p> <p>The <code>scr_set()</code> routine is a combination of <code>scr_restore()</code> and <code>scr_init()</code>. It tells the program that the information in <i>filename</i> is what is currently on the screen, and also what the program wants on the screen. This can be thought of as a screen inheritance function.</p> <p>To read (write) a window from (to) a file, use the <code>getwin()</code> and <code>putwin()</code> routines (see <code>curs_util(3CURSES)</code>).</p> | | | | |
| RETURN VALUES | All routines return the integer ERR upon failure and OK upon success. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_initscr(3CURSES)</code> , <code>curs_refresh(3CURSES)</code> , <code>curs_util(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>system(3C)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . | | | | |

`scr_dump(3CURSES)`

Note that `scr_init()`, `scr_set()`, and `scr_restore()` may be macros.

| NAME | scr_dump, scr_init, scr_restore, scr_set – write screen contents to/from a file | | | | | | |
|----------------------|--|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int scr_dump(const char *filename); int scr_init(const char *filename); int scr_restore(const char *filename); int scr_set(const char *filename);</pre> | | | | | | |
| PARAMETERS | <i>filename</i> Is a pointer to the file in which screen contents are written. | | | | | | |
| DESCRIPTION | <p>These function perform input/output functions on a screen basis.</p> <p>The <code>scr_dump()</code> function writes the contents of the virtual screen, <code>curscr</code>, to <i>filename</i>.</p> <p>The <code>scr_restore()</code> function reads the contents of <i>filename</i> from <code>curscr</code> (which must have been written with <code>scr_dump()</code>). The next refresh operation restores the screen to the way it looks in <i>filename</i>.</p> <p>The <code>scr_init()</code> function reads the contents of <i>filename</i> and uses those contents to initialize the X/Open Curses data structures to what is actually on screen. The next refresh operation bases its updates on this data, unless the terminal has been written to since <i>filename</i> was saved or the terminfo capabilities <code>rmcup</code> and <code>nrrmc</code> are defined for the current terminal.</p> <p>The <code>scr_set()</code> function combines <code>scr_restore()</code> and <code>scr_init()</code>. It informs the program that the contents of the file <i>filename</i> are what is currently on the screen and that the program wants those contents on the screen.</p> | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>delscreen(3XCURSES)</code> , <code>doupdate(3XCURSES)</code> , <code>endwin(3XCURSES)</code> , <code>getwin(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

scr_init(3CURSES)

| NAME | <code>scr_dump</code> , <code>scr_dump</code> , <code>scr_restore</code> , <code>scr_init</code> , <code>scr_set</code> – read (write) a curses screen from (to) a file | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int scr_dump(char *filename); int scr_restore(char *filename); int scr_init(char *filename); int scr_set(char *filename);</pre> | | | | |
| DESCRIPTION | <p>With the <code>scr_dump()</code> routine, the current contents of the virtual screen are written to the file <i>filename</i>.</p> <p>With the <code>scr_restore()</code> routine, the virtual screen is set to the contents of <i>filename</i>, which must have been written using <code>scr_dump()</code>. The next call to <code>doupdate()</code> restores the screen to the way it looked in the dump file.</p> <p>With the <code>scr_init()</code> routine, the contents of <i>filename</i> are read in and used to initialize the curses data structures about what the terminal currently has on its screen. If the data is determined to be valid, curses bases its next update of the screen on this information rather than clearing the screen and starting from scratch. <code>scr_init()</code> is used after <code>initscr()</code> or a <code>system(3C)</code> call to share the screen with another process which has done a <code>scr_dump()</code> after its <code>endwin()</code> call. The data is declared invalid if the time-stamp of the tty is old or the terminfo capabilities <code>rmcup()</code> and <code>nrrmc()</code> exist.</p> <p>The <code>scr_set()</code> routine is a combination of <code>scr_restore()</code> and <code>scr_init()</code>. It tells the program that the information in <i>filename</i> is what is currently on the screen, and also what the program wants on the screen. This can be thought of as a screen inheritance function.</p> <p>To read (write) a window from (to) a file, use the <code>getwin()</code> and <code>putwin()</code> routines (see <code>curs_util(3CURSES)</code>).</p> | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and <code>OK</code> upon success. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_initscr(3CURSES)</code> , <code>curs_refresh(3CURSES)</code> , <code>curs_util(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>system(3C)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . | | | | |

`scr_init(3CURSES)`

Note that `scr_init()`, `scr_set()`, and `scr_restore()` may be macros.

scr_init(3XCURSES)

| NAME | scr_dump, scr_init, scr_restore, scr_set – write screen contents to/from a file | | | | | | |
|----------------------|--|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int scr_dump(const char *filename); int scr_init(const char *filename); int scr_restore(const char *filename); int scr_set(const char *filename);</pre> | | | | | | |
| PARAMETERS | <i>filename</i> Is a pointer to the file in which screen contents are written. | | | | | | |
| DESCRIPTION | <p>These function perform input/output functions on a screen basis.</p> <p>The <code>scr_dump()</code> function writes the contents of the virtual screen, <code>curscr</code>, to <i>filename</i>.</p> <p>The <code>scr_restore()</code> function reads the contents of <i>filename</i> from <code>curscr</code> (which must have been written with <code>scr_dump()</code>). The next refresh operation restores the screen to the way it looks in <i>filename</i>.</p> <p>The <code>scr_init()</code> function reads the contents of <i>filename</i> and uses those contents to initialize the X/Open Curses data structures to what is actually on screen. The next refresh operation bases its updates on this data, unless the terminal has been written to since <i>filename</i> was saved or the terminfo capabilities <code>rmcup</code> and <code>nrrmc</code> are defined for the current terminal.</p> <p>The <code>scr_set()</code> function combines <code>scr_restore()</code> and <code>scr_init()</code>. It informs the program that the contents of the file <i>filename</i> are what is currently on the screen and that the program wants those contents on the screen.</p> | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Standard</td></tr><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>delscreen(3XCURSES)</code> , <code>doupdate(3XCURSES)</code> , <code>endwin(3XCURSES)</code> , <code>getwin(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

| NAME | curs_scroll, scroll, sclr, wscrl – scroll a curses window | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int scroll(WINDOW *win); int sclr(int n); int wscrl(WINDOW *win, int n);</pre> | | | | |
| DESCRIPTION | <p>With the <code>scroll()</code> routine, the window is scrolled up one line. This involves moving the lines in the window data structure. As an optimization, if the scrolling region of the window is the entire screen, the physical screen is scrolled at the same time.</p> <p>With the <code>sclr()</code> and <code>wscrl()</code> routines, for positive n scroll the window up n lines (line $i+n$ becomes i); otherwise scroll the window down n lines. This involves moving the lines in the window character image structure. The current cursor position is not changed.</p> <p>For these functions to work, scrolling must be enabled via <code>scrollok()</code>.</p> | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_outopts(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that <code>sclr()</code> and <code>scroll()</code> may be macros.</p> | | | | |

srl(3XCURSES)

| NAME | srl, scroll, wscr – scroll a window | | | | | | |
|----------------------|--|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int srl(int <i>n</i>); int scroll(WINDOW *<i>win</i>); int wscr(WINDOW *<i>win</i>, int <i>n</i>);</pre> | | | | | | |
| PARAMETERS | <p><i>n</i> number and direction of lines to scroll</p> <p><i>win</i> pointer to the window in which to scroll</p> | | | | | | |
| DESCRIPTION | <p>The <code>scroll()</code> function scrolls the window <i>win</i> up one line. The current cursor position is not changed.</p> <p>The <code>srl()</code> and <code>wscr()</code> functions scroll the window <code>stdscr</code> or <i>win</i> up or down <i>n</i> lines, where <i>n</i> is a positive (scroll up) or negative (scroll down) integer.</p> <p>The <code>scrollok(3XCURSES)</code> function must be enabled for these functions to work.</p> | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Standard</td></tr><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>clearok(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

| NAME | curl_scroll, scroll, scl, wscrl – scroll a curses window | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int scroll(WINDOW *win); int scl(int n); int wscrl(WINDOW *win, int n);</pre> | | | | |
| DESCRIPTION | <p>With the scroll() routine, the window is scrolled up one line. This involves moving the lines in the window data structure. As an optimization, if the scrolling region of the window is the entire screen, the physical screen is scrolled at the same time.</p> <p>With the scl() and wscrl() routines, for positive <i>n</i> scroll the window up <i>n</i> lines (line <i>i+n</i> becomes <i>i</i>); otherwise scroll the window down <i>n</i> lines. This involves moving the lines in the window character image structure. The current cursor position is not changed.</p> <p>For these functions to work, scrolling must be enabled via scrollok().</p> | | | | |
| RETURN VALUES | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curl_outopts(3CURSES), curses(3CURSES), attributes(5) | | | | |
| NOTES | <p>The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.</p> <p>Note that scl() and scroll() may be macros.</p> | | | | |

scroll(3XCURSES)

NAME | scl, scroll, wscrl – scroll a window

SYNOPSIS | `cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \`
| `-R /usr/xpg4/lib -lcurses [library...]`

| `c89 [flag...] file... -lcurses [library...]`

| `#include <curses.h>`
 | `int scl(int n);`
 | `int scroll(WINDOW *win);`
 | `int wscrl(WINDOW *win, int n);`

PARAMETERS | *n* number and direction of lines to scroll
 | *win* pointer to the window in which to scroll

DESCRIPTION | The scroll() function scrolls the window *win* up one line. The current cursor position is not changed.

| The scl() and wscrl() functions scroll the window *stdscr* or *win* up or down *n* lines, where *n* is a positive (scroll up) or negative (scroll down) integer.

| The scrollok(3XCURSES) function must be enabled for these functions to work.

RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR.

ERRORS | None.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | clearok(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

| | |
|--------------------|--|
| NAME | curs_ouptoks, clearok, idlok, idcok, immedok, leaveok, setscreg, wsetscreg, scrollok, nl, nonl – curses terminal output option control routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int clearok(WINDOW *win, bool bf); int idlok(WINDOW *win, bool bf); void idcok(WINDOW *win, bool bf); void immedok(WINDOW *win, bool bf); int leaveok(WINDOW *win, bool bf); int setscreg(int top, int bot); int wsetscreg(WINDOW *win, int top, int bot); int scrollok(WINDOW *win, bool bf); int nl(void); int nonl(void);</pre> |
| DESCRIPTION | <p>These routines set options that deal with output within curses. All options are initially FALSE, unless otherwise stated. It is not necessary to turn these options off before calling <code>endwin()</code>.</p> <p>With the <code>clearok()</code> routine, if enabled (<i>bf</i> is TRUE), the next call to <code>wrefresh()</code> with this window will clear the screen completely and redraw the entire screen from scratch. This is useful when the contents of the screen are uncertain, or in some cases for a more pleasing visual effect. If the <i>win</i> argument to <code>clearok()</code> is the global variable <code>curscr()</code>, the next call to <code>wrefresh()</code> with any window causes the screen to be cleared and repainted from scratch.</p> <p>With the <code>idlok()</code> routine, if enabled (<i>bf</i> is TRUE), <code>curses</code> considers using the hardware insert/delete line feature of terminals so equipped. If disabled (<i>bf</i> is FALSE), <code>curses</code> very seldom uses this feature. (The insert/delete character feature is always considered.) This option should be enabled only if the application needs insert/delete line, for example, for a screen editor. It is disabled by default because insert/delete line tends to be visually annoying when used in applications where it isn't really needed. If insert/delete line cannot be used, <code>curses</code> redraws the changed portions of all lines.</p> <p>With the <code>idcok()</code> routine, if enabled (<i>bf</i> is TRUE), <code>curses</code> considers using the hardware insert/delete character feature of terminals so equipped. This is enabled by default.</p> <p>With the <code>immedok()</code> routine, if enabled (<i>bf</i> is TRUE), any change in the window image, such as the ones caused by <code>waddch()</code>, <code>wclrtoeol()</code>, <code>wscrl()</code>, etc., automatically cause a call to <code>wrefresh()</code>. However, it may degrade the performance considerably, due to repeated calls to <code>wrefresh()</code>. It is disabled by default.</p> |

scrollok(3CURSES)

Normally, the hardware cursor is left at the location of the window cursor being refreshed. The `leaveok()` option allows the cursor to be left wherever the update happens to leave it. It is useful for applications where the cursor is not used, since it reduces the need for cursor motions. If possible, the cursor is made invisible when this option is enabled.

The `setscrreg()` and `wsetscrreg()` routines allow the application programmer to set a software scrolling region in a window. *top* and *bot* are the line numbers of the top and bottom margin of the scrolling region. (Line 0 is the top line of the window.) If this option and `scrollok()` are enabled, an attempt to move off the bottom margin line causes all lines in the scrolling region to scroll up one line. Only the text of the window is scrolled. (Note that this has nothing to do with the use of a physical scrolling region capability in the terminal, like that in the VT100. If `idlok()` is enabled and the terminal has either a scrolling region or insert/delete line capability, they will probably be used by the output routines.)

The `scrollok()` option controls what happens when the cursor of a window is moved off the edge of the window or scrolling region, either as a result of a newline action on the bottom line, or typing the last character of the last line. If disabled, (*bf* is FALSE), the cursor is left on the bottom line. If enabled, (*bf* is TRUE), `wrefresh()` is called on the window, and the physical terminal and window are scrolled up one line. (Note that in order to get the physical scrolling effect on the terminal, it is also necessary to call `idlok()`.)

The `nl()` and `nonl()` routines control whether newline is translated into carriage return and linefeed on output, and whether return is translated into newline on input. Initially, the translations do occur. By disabling these translations using `nonl()`, `curses` is able to make better use of the linefeed capability, resulting in faster cursor motion.

RETURN VALUES `setscrreg()` and `wsetscrreg()` return OK upon success and ERR upon failure. All other routines that return an integer always return OK.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_addch(3CURSES)`, `curs_clear(3CURSES)`, `curs_initscr(3CURSES)`, `curs_refresh(3CURSES)`, `curs_scroll(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `clearok()`, `leaveok()`, `scrollok()`, `idcok()`, `nl()`, `nonl()`, and `setscrreg()` may be macros.

`scrollok(3CURSES)`

The `immedok()` routine is useful for windows that are used as terminal emulators.

scrollok(3XCURSES)

| | |
|--------------------|---|
| NAME | clearok, idlok, leaveok, scrollok, setscrreg, wsetscreg – terminal output control functions |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int clearok(WINDOW *win, bool bf); int idlok(WINDOW *win, bool bf); int leaveok(WINDOW *win, bool bf); int scrollok(WINDOW *win, bool bf); int setscreg(int top, int bot); int wsetscreg(WINDOW *win, int top, int bot);</pre> |
| DESCRIPTION | <p>These functions set options that deal with the output within Curses functions.</p> <p>The <code>clearok()</code> function assigns the value of <i>bf</i> to an internal flag in the specified window that governs clearing of the screen during a refresh. If, during a refresh operation on the specified window, the flag in <i>curscr</i> is TRUE or the flag in the specified window is TRUE, <code>clearok()</code> clears the screen, redraws it in its entirety, and sets the flag to FALSE in <i>curscr</i> and in the specified window. The initial state is unspecified</p> <p>The <code>idlok()</code> function specifies whether the implementation may use the hardware insert-line, delete-line, and scroll features of terminals so equipped. If <i>bf</i> is TRUE, use of these features is enabled. If <i>bf</i> is FALSE, use of these features is disabled and lines are instead redrawn as required. The initial state is FALSE.</p> <p>The <code>leaveok()</code> function controls the cursor position after a refresh operation. If <i>bf</i> is TRUE, refresh operations on the specified window may leave the terminal's cursor at an arbitrary position. If <i>bf</i> is FALSE, then at the end of any refresh operation, the terminal's cursor is positioned at the cursor position contained in the specified window. The initial state is FALSE.</p> <p>The <code>scrollok()</code> function controls the use of scrolling. If <i>bf</i> is TRUE, then scrolling is enabled for the specified window. If <i>bf</i> is FALSE, scrolling is disabled for the specified window. The initial state is FALSE.</p> <p>The <code>setscreg()</code> and <code>wsetscreg()</code> functions define a software scrolling region in the current or specified window. The <i>top</i> and <i>bottom</i> arguments are the line numbers of the first and last line defining the scrolling region. (Line 0 is the top line of the window.) If this option and <code>scrollok()</code> are enabled, an attempt to move off the last line of the margin causes all lines in the scrolling region to scroll one line in the</p> |

scrollok(3XCURSES)

direction of the first line. Only characters in the window are scrolled. If a software scrolling region is set and `scrollok()` is not enabled, an attempt to move off the last line of the margin does not reposition any lines in the scrolling region.

PARAMETERS

| | |
|------------|---|
| <i>win</i> | Is a pointer to a window. |
| <i>bf</i> | Is a Boolean expression. |
| <i>top</i> | Is the top line of the scrolling region (top of the window is line 0). |
| <i>bot</i> | Is the bottom line of the scrolling region (top of the window is line 0). |

RETURN VALUES Upon successful completion, the `setscrreg()` and `wsetscrreg()` functions return OK. Otherwise, they return ERR.

The other functions always return OK.

ERRORS No errors are defined.

USAGE The only reason to enable the `idlok()` feature is to use scrolling to achieve the visual effect of motion of a partial window, such as for a screen editor. In other cases, the feature can be visually annoying.

The `leaveok()` option provides greater efficiency for applications that do not use the cursor.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `bkgdset(3XCURSES)`, `clear(3XCURSES)`, `doupdate(3XCURSES)`, `libcurses(3XCURSES)`, `scr1(3XCURSES)`, `attributes(5)`, `standards(5)`

scr_restore(3CURSES)

| NAME | <code> curs_scr_dump, scr_dump, scr_restore, scr_init, scr_set</code> – read (write) a curses screen from (to) a file | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int scr_dump(char *filename); int scr_restore(char *filename); int scr_init(char *filename); int scr_set(char *filename);</pre> | | | | |
| DESCRIPTION | <p>With the <code>scr_dump()</code> routine, the current contents of the virtual screen are written to the file <i>filename</i>.</p> <p>With the <code>scr_restore()</code> routine, the virtual screen is set to the contents of <i>filename</i>, which must have been written using <code>scr_dump()</code>. The next call to <code>doupdate()</code> restores the screen to the way it looked in the dump file.</p> <p>With the <code>scr_init()</code> routine, the contents of <i>filename</i> are read in and used to initialize the curses data structures about what the terminal currently has on its screen. If the data is determined to be valid, curses bases its next update of the screen on this information rather than clearing the screen and starting from scratch. <code>scr_init()</code> is used after <code>initscr()</code> or a <code>system(3C)</code> call to share the screen with another process which has done a <code>scr_dump()</code> after its <code>endwin()</code> call. The data is declared invalid if the time-stamp of the tty is old or the terminfo capabilities <code>rmcup()</code> and <code>nrrmc()</code> exist.</p> <p>The <code>scr_set()</code> routine is a combination of <code>scr_restore()</code> and <code>scr_init()</code>. It tells the program that the information in <i>filename</i> is what is currently on the screen, and also what the program wants on the screen. This can be thought of as a screen inheritance function.</p> <p>To read (write) a window from (to) a file, use the <code>getwin()</code> and <code>putwin()</code> routines (see <code>curs_util(3CURSES)</code>).</p> | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and <code>OK</code> upon success. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_initscr(3CURSES)</code> , <code>curs_refresh(3CURSES)</code> , <code>curs_util(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>system(3C)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . | | | | |

`scr_restore(3CURSES)`

Note that `scr_init()`, `scr_set()`, and `scr_restore()` may be macros.

scr_restore(3XCURSES)

| NAME | scr_dump, scr_init, scr_restore, scr_set – write screen contents to/from a file | | | | | | |
|----------------------|--|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int scr_dump(const char *filename); int scr_init(const char *filename); int scr_restore(const char *filename); int scr_set(const char *filename);</pre> | | | | | | |
| PARAMETERS | <i>filename</i> Is a pointer to the file in which screen contents are written. | | | | | | |
| DESCRIPTION | <p>These function perform input/output functions on a screen basis.</p> <p>The <code>scr_dump()</code> function writes the contents of the virtual screen, <code>curscr</code>, to <i>filename</i>.</p> <p>The <code>scr_restore()</code> function reads the contents of <i>filename</i> from <code>curscr</code> (which must have been written with <code>scr_dump()</code>). The next refresh operation restores the screen to the way it looks in <i>filename</i>.</p> <p>The <code>scr_init()</code> function reads the contents of <i>filename</i> and uses those contents to initialize the X/Open Curses data structures to what is actually on screen. The next refresh operation bases its updates on this data, unless the terminal has been written to since <i>filename</i> was saved or the terminfo capabilities <code>rmcup</code> and <code>nrrmc</code> are defined for the current terminal.</p> <p>The <code>scr_set()</code> function combines <code>scr_restore()</code> and <code>scr_init()</code>. It informs the program that the contents of the file <i>filename</i> are what is currently on the screen and that the program wants those contents on the screen.</p> | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Standard</td></tr><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>delscreen(3XCURSES)</code> , <code>doupdate(3XCURSES)</code> , <code>endwin(3XCURSES)</code> , <code>getwin(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

| NAME | curs_scr_dump, scr_dump, scr_restore, scr_init, scr_set – read (write) a curses screen from (to) a file | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int scr_dump(char *filename); int scr_restore(char *filename); int scr_init(char *filename); int scr_set(char *filename);</pre> | | | | |
| DESCRIPTION | <p>With the <code>scr_dump()</code> routine, the current contents of the virtual screen are written to the file <i>filename</i>.</p> <p>With the <code>scr_restore()</code> routine, the virtual screen is set to the contents of <i>filename</i>, which must have been written using <code>scr_dump()</code>. The next call to <code>doupdate()</code> restores the screen to the way it looked in the dump file.</p> <p>With the <code>scr_init()</code> routine, the contents of <i>filename</i> are read in and used to initialize the curses data structures about what the terminal currently has on its screen. If the data is determined to be valid, curses bases its next update of the screen on this information rather than clearing the screen and starting from scratch. <code>scr_init()</code> is used after <code>initscr()</code> or a <code>system(3C)</code> call to share the screen with another process which has done a <code>scr_dump()</code> after its <code>endwin()</code> call. The data is declared invalid if the time-stamp of the tty is old or the terminfo capabilities <code>rmcup()</code> and <code>nrrmc()</code> exist.</p> <p>The <code>scr_set()</code> routine is a combination of <code>scr_restore()</code> and <code>scr_init()</code>. It tells the program that the information in <i>filename</i> is what is currently on the screen, and also what the program wants on the screen. This can be thought of as a screen inheritance function.</p> <p>To read (write) a window from (to) a file, use the <code>getwin()</code> and <code>putwin()</code> routines (see <code>curs_util(3CURSES)</code>).</p> | | | | |
| RETURN VALUES | All routines return the integer ERR upon failure and OK upon success. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_initscr(3CURSES)</code> , <code>curs_refresh(3CURSES)</code> , <code>curs_util(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>system(3C)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . | | | | |

`scr_set(3CURSES)`

Note that `scr_init()`, `scr_set()`, and `scr_restore()` may be macros.

NAME | scr_dump, scr_init, scr_restore, scr_set – write screen contents to/from a file

SYNOPSIS |

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

int scr_dump(const char *filename);
int scr_init(const char *filename);
int scr_restore(const char *filename);
int scr_set(const char *filename);
```

PARAMETERS | *filename* Is a pointer to the file in which screen contents are written.

DESCRIPTION | These function perform input/output functions on a screen basis.

The `scr_dump()` function writes the contents of the virtual screen, `curscr`, to *filename*.

The `scr_restore()` function reads the contents of *filename* from `curscr` (which must have been written with `scr_dump()`). The next refresh operation restores the screen to the way it looks in *filename*.

The `scr_init()` function reads the contents of *filename* and uses those contents to initialize the X/Open Curses data structures to what is actually on screen. The next refresh operation bases its updates on this data, unless the terminal has been written to since *filename* was saved or the terminfo capabilities `rmcup` and `nrrmc` are defined for the current terminal.

The `scr_set()` function combines `scr_restore()` and `scr_init()`. It informs the program that the contents of the file *filename* are what is currently on the screen and that the program wants those contents on the screen.

RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR.

ERRORS | None.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | `delscreen(3XCURSES)`, `doupdate(3XCURSES)`, `endwin(3XCURSES)`, `getwin(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

setcchar(3XCURSES)

NAME setcchar – set a cchar_t type character from a wide character and rendition

SYNOPSIS `cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [library...]`

`c89 [flag...] file... -lcurses [library...]`

```
#include <curses.h>

int setcchar(cchar_t *wcv, const wchar_t *wch, const attr_t attrs,
             short color_pair, const void *opts);
```

PARAMETERS

| | |
|-------------------|---|
| <i>wcv</i> | Is a pointer to a location where a cchar_t character (and its rendition) can be stored. |
| <i>wch</i> | Is a pointer to a wide character. |
| <i>attrs</i> | Is the set of attributes to apply to <i>wch</i> in creating <i>wcv</i> . |
| <i>color_pair</i> | Is the color pair to apply to <i>wch</i> in creating <i>wcv</i> . |
| <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. |

DESCRIPTION The setcchar() function takes the wide character pointed to by *wch*, combines it with the attributes indicated by *attrs* and the color pair indicated by *color_pair* and stores the result in the object pointed to by *wcv*.

RETURN VALUES On success, the setcchar() function returns OK. Otherwise, it returns ERR.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO attroff(3XCURSES), can_change_color(3XCURSES), getcchar(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

| NAME | form_page, set_form_page, set_current_field, current_field, field_index – set forms current page and field | | | | | | | | | | | | |
|----------------------|--|----------------|-------------------------------------|----------------|---------------|----------------|---------------------------|-------------|--|-----------------|---------------------------------|------------------|--------------------------------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_form_page(FORM *form, int page); int form_page(FORM *form); int set_current_field(FORM *form, FIELD *field); FIELD *current_field(FORM*form); int field_index(FIELD *field);</pre> | | | | | | | | | | | | |
| DESCRIPTION | <p>set_form_page() sets the page number of <i>form</i> to <i>page</i>. form_page() returns the current page number of <i>form</i>.</p> <p>set_current_field() sets the current field of <i>form</i> to <i>field</i>. current_field() returns a pointer to the current field of <i>form</i>.</p> <p>field_index() returns the index in the field pointer array of <i>field</i>.</p> | | | | | | | | | | | | |
| RETURN VALUES | <p>form_page() returns -1 on error.</p> <p>current_field() returns NULL on error.</p> <p>field_index() returns -1 on error.</p> <p>set_form_page() and set_current_field() return one of the following:</p> <table border="0"> <tr> <td style="padding-right: 20px;">E_OK</td> <td>The function returned successfully.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> <tr> <td>E_BAD_ARGUMENT</td> <td>An argument is incorrect.</td> </tr> <tr> <td>E_BAD_STATE</td> <td>The routine was called from an initialization or termination function.</td> </tr> <tr> <td>E_INVALID_FIELD</td> <td>The field contents are invalid.</td> </tr> <tr> <td>E_REQUEST_DENIED</td> <td>The form driver request failed</td> </tr> </table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An argument is incorrect. | E_BAD_STATE | The routine was called from an initialization or termination function. | E_INVALID_FIELD | The field contents are invalid. | E_REQUEST_DENIED | The form driver request failed |
| E_OK | The function returned successfully. | | | | | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | | | | | |
| E_BAD_ARGUMENT | An argument is incorrect. | | | | | | | | | | | | |
| E_BAD_STATE | The routine was called from an initialization or termination function. | | | | | | | | | | | | |
| E_INVALID_FIELD | The field contents are invalid. | | | | | | | | | | | | |
| E_REQUEST_DENIED | The form driver request failed | | | | | | | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | | | | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | | | | | | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | | | | | | | |
| MT-Level | Unsafe | | | | | | | | | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | | | | | | | | | |

set_current_field(3CURSES)

NOTES | The header `<form.h>` automatically includes the headers `<eti.h>` and `<curses.h>`.

set_current_item(3CURSES)

NAME menu_item_current, set_current_item, current_item, set_top_row, top_row, item_index
– set and get current menus items

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

int set_current_item(MENU *menu, ITEM *item);
ITEM *current_item(MENU *menu);
int set_top_row(MENU *menu, int row);
int top_row(MENU *menu);
int item_index(ITEM *item);
```

DESCRIPTION

The current item of a menu is the item where the cursor is currently positioned. `set_current_item()` sets the current item of *menu* to *item*. `current_item()` returns a pointer to the the current item in *menu*.

`set_top_row()` sets the top row of *menu* to *row*. The left-most item on the new top row becomes the current item. `top_row()` returns the number of the menu row currently displayed at the top of *menu*.

`item_index()` returns the index to the *item* in the item pointer array. The value of this index ranges from 0 through *N*-1, where *N* is the total number of items connected to the menu.

RETURN VALUES

`current_item()` returns NULL on error.

`top_row()` and `index_item()` return -1 on error.

`set_current_item()` and `set_top_row()` return one of the following:

| | |
|-----------------|--|
| E_OK | The routine returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. |
| E_BAD_STATE | The routine was called from an initialization or termination function. |
| E_NOT_CONNECTED | No items are connected to the menu. |

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3CURSES)`, `menus(3CURSES)`, `attributes(5)`

set_current_item(3CURSES)

NOTES | The header `<menu.h>` automatically includes the headers `<eti.h>` and `<curses.h>`.

| | |
|--------------------|--|
| NAME | curs_terminfo, setupterm, setterm, set_curterm, del_curterm, restartterm, tparm, tputs, putp, vidputs, vidattr, mvcur, tigetflag, tigetnum, tigetstr – curses interfaces to terminfo database |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> #include <term.h> int setupterm(char *term, int fildes, int *errret); int setterm(char *term); int set_curterm(TERMINAL *nterm); int del_curterm(TERMINAL *oterm); int restartterm(char *term, int fildes, int *errret); char *tparm(char *str, long int p1, long int p2, long int p3, long int p4, long int p5, long int p6, long int p7, long int p8, long int p9); int tputs(char *str, int affcnt, int (*putc)(char)); int putp(char *str); int vidputs(chtype attrs, int (*putc)(char)); int vidattr(chtype attrs); int mvcur(int oldrow, int oldcol, int newrow, int newcol); int tigetflag(char *capname); int tigetnum(char *capname); char *tigetstr(char *capname);</pre> |
| DESCRIPTION | <p>These low-level routines must be called by programs that have to deal directly with the <i>terminfo</i> database to handle certain terminal capabilities, such as programming function keys. For all other functionality, curses routines are more suitable and their use is recommended.</p> <p>Initially, <code>setupterm()</code> should be called. Note that <code>setupterm()</code> is automatically called by <code>initscr()</code> and <code>newterm()</code>. This defines the set of terminal-dependent variables (listed in <code>terminfo(4)</code>). The <i>terminfo</i> variables <code>lines</code> and <code>columns</code> are initialized by <code>setupterm()</code> as follows: If <code>use_env(FALSE)</code> has been called, values for <code>lines</code> and <code>columns</code> specified in <i>terminfo</i> are used. Otherwise, if the environment variables <code>LINES</code> and <code>COLUMNS</code> exist, their values are used. If these environment variables do not exist and the program is running in a window, the current window size is used. Otherwise, if the environment variables do not exist, the values for <code>lines</code> and <code>columns</code> specified in the <i>terminfo</i> database are used.</p> <p>The headers <code><curses.h></code> and <code><term.h></code> should be included (in this order) to get the definitions for these strings, numbers, and flags. Parameterized strings should be passed through <code>tparm()</code> to instantiate them. All <i>terminfo</i> strings (including the output</p> |

set_curterm(3CURSES)

of `tparm()` should be printed with `tputs()` or `putp()`. Call the `reset_shell_mode()` routine to restore the tty modes before exiting (see `curls_kernel(3CURSES)`). Programs which use cursor addressing should output `enter_ca_mode` upon startup and should output `exit_ca_mode` before exiting. Programs desiring shell escapes should call `reset_shell_mode` and output `exit_ca_mode` before the shell is called and should output `enter_ca_mode` and call `reset_prog_mode` after returning from the shell.

The `setupterm()` routine reads in the *terminfo* database, initializing the *terminfo* structures, but does not set up the output virtualization structures used by *curses*. The terminal type is the character string *term*; if *term* is null, the environment variable `TERM` is used. All output is to file descriptor *fildev* which is initialized for output. If *errret* is not null, then `setupterm()` returns `OK` or `ERR` and stores a status value in the integer pointed to by *errret*. A status of 1 in *errret* is normal, 0 means that the terminal could not be found, and -1 means that the *terminfo* database could not be found. If *errret* is null, `setupterm()` prints an error message upon finding an error and exits. Thus, the simplest call is:

```
setupterm((char *)0, 1, (int *)0);
```

which uses all the defaults and sends the output to `stdout`.

The `setterm()` routine is being replaced by `setupterm()`. The call:

```
setupterm(term, 1, (int *)0)
```

provides the same functionality as `setterm(term)`. The `setterm()` routine is included here for compatibility and is supported at Level 2.

The `set_curterm()` routine sets the variable `cur_term` to *nterm*, and makes all of the *terminfo* boolean, numeric, and string variables use the values from *nterm*.

The `del_curterm()` routine frees the space pointed to by *oterm* and makes it available for further use. If *oterm* is the same as `cur_term`, references to any of the *terminfo* boolean, numeric, and string variables thereafter may refer to invalid memory locations until another `setupterm()` has been called.

The `restartterm()` routine is similar to `setupterm()` and `initscr()`, except that it is called after restoring memory to a previous state. It assumes that the windows and the input and output options are the same as when memory was saved, but the terminal type and baud rate may be different.

The `tparm()` routine instantiates the string *str* with parameters *pi*. A pointer is returned to the result of *str* with the parameters applied.

The `tputs()` routine applies padding information to the string *str* and outputs it. The *str* must be a *terminfo* string variable or the return value from `tparm()`, `tgetstr()`, or `tgoto()`. *affcnt* is the number of lines affected, or 1 if not applicable. *putc* is a `putchar()`-like routine to which the characters are passed, one at a time.

set_curterm(3CURSES)

The `putp()` routine calls `tputs(str, 1, putchar)`. Note that the output of `putpA()` always goes to `stdout`, not to the *files* specified in `setupterm()`.

The `vidputs()` routine displays the string on the terminal in the video attribute mode *attrs*, which is any combination of the attributes listed in `curses(3CURSES)`. The characters are passed to the `putchar()`-like routine `putc()`.

The `vidattr()` routine is like the `vidputs()` routine, except that it outputs through `putchar()`.

The `mvcur()` routine provides low-level cursor motion.

The `tigetflag()`, `tigetnum()` and `tigetstr()` routines return the value of the capability corresponding to the *terminfo capname* passed to them, such as `xenl`.

With the `tigetflag()` routine, the value `-1` is returned if *capname* is not a boolean capability.

With the `tigetnum()` routine, the value `-2` is returned if *capname* is not a numeric capability.

With the `tigetstr()` routine, the value `(char *)-1` is returned if *capname* is not a string capability.

The *capname* for each capability is given in the table column entitled *capname* code in the capabilities section of `terminfo(4)`.

```
char *boolnames, *boolcodes, *boolfnames
char *numnames, *numcodes, *numfnames
char *strnames, *strcodes, *strfnames
```

These null-terminated arrays contain the *capnames*, the *termcap* codes, and the full C names, for each of the *terminfo* variables.

RETURN VALUES

All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

Routines that return pointers always return `NULL` on error.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_initscr(3CURSES)`, `curs_kernel(3CURSES)`, `curs_termcap(3CURSES)`, `curses(3CURSES)`, `putc(3C)`, `terminfo(4)`, `attributes(5)`

set_curterm(3CURSES)

NOTES | The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

The `setupterm()` routine should be used in place of `setterm()`.

Note that `vidattr()` and `vidputs()` may be macros.

| NAME | del_curterm, restartterm, set_curterm, setupterm – interfaces to the terminfo database | | | | | | | | |
|--------------------|---|-------|-------------|---|--------|---|-----------------------------|----|--------------------------------------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <term.h> int del_curterm(TERMINAL *oterm); int restartterm(char *term, int fildes, int *errret); TERMINAL *set_curterm(TERMINAL *nterm); int setupterm(char *term, int fildes, int *errret);</pre> | | | | | | | | |
| DESCRIPTION | <p>Within X/Open Curses, the <code>setupterm()</code> function is automatically called by the <code>initscr(3XC)</code> and <code>newterm(3XC)</code> functions. This function can be also be used outside of X/Open Curses when a program has to deal directly with the <code>terminfo</code> database to handle certain terminal capabilities. The use of appropriate X/Open Curses functions is recommended in all other situations.</p> <p>The <code>setupterm()</code> function loads terminal-dependent variables for the <code>terminfo</code> layer of X/Open Curses. The <code>setupterm()</code> function initializes the <code>terminfo</code> variables <code>lines</code> and <code>columns</code> such that if <code>use_env(FALSE)</code> has been called, the <code>terminfo</code> values assigned in the database are used regardless of the environmental variables <code>LINES</code> and <code>COLUMNS</code> or the program's window dimensions; when <code>use_env(TRUE)</code> has been called, which is the default, the environment variables <code>LINES</code> and <code>COLUMNS</code> are used, if they exist. If the environment variables do not exist and the program is running in a window, the current window size is used.</p> <p>The <code>term</code> parameter of <code>setupterm()</code> specifies the terminal; if null, terminal type is taken from the <code>TERM</code> environment variable. All output is sent to <code>fildes</code> which is initialized for output. If <code>errret</code> is not null, <code>OK</code> or <code>ERR</code> is returned and a status value is stored in the integer pointed to by <code>errret</code>. The following status values may be returned:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Value</th> <th style="text-align: left;">Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Normal</td> </tr> <tr> <td>0</td> <td>Terminal could not be found</td> </tr> <tr> <td>-1</td> <td>terminfo database could not be found</td> </tr> </tbody> </table> <p>If <code>errret</code> is null, an error message is printed, and the <code>setupterm()</code> function calls the <code>exit()</code> function with a non-zero parameter.</p> | Value | Description | 1 | Normal | 0 | Terminal could not be found | -1 | terminfo database could not be found |
| Value | Description | | | | | | | | |
| 1 | Normal | | | | | | | | |
| 0 | Terminal could not be found | | | | | | | | |
| -1 | terminfo database could not be found | | | | | | | | |

set_curterm(3XCURSES)

The `set_curterm()` function sets the `cur_term` variable to `nterm`. The values from `nterm` as well as other state information for the terminal are used by X/Open Curses functions such as `beep(3XCURSES)`, `flash(3XCURSES)`, `mvcur(3XCURSES)`, `tigetflag(3XCURSES)`, `tigetstr(3XCURSES)`, and `tigetnum(3XCURSES)`.

The `del_curterm()` function frees the space pointed to by `oterm`. If `oterm` and the `cur_term` variable are the same, all Boolean, numeric, or string terminfo variables will refer to invalid memory locations until you call `setupterm()` and specify a new terminal type.

The `restartterm()` function assumes that a call to `setupterm()` has already been made (probably from `initscr()` or `newterm()`). It allows you to specify a new terminal type in `term` and updates the data returned by `baudrate(3XCURSES)` based on `fildev`. Other information created by the `initscr()`, `newterm()`, and `setupterm()` functions is preserved.

| | | |
|-------------------|---------------|---|
| PARAMETERS | <i>oterm</i> | Is the terminal type for which to free space. |
| | <i>term</i> | Is the terminal type for which variables are set. |
| | <i>fildev</i> | Is a file descriptor initialized for output. |
| | <i>errret</i> | Is a pointer to an integer in which the status value is stored. |
| | <i>nterm</i> | Is the new terminal to become the current terminal. |

RETURN VALUES On success, the `set_curterm()` function returns the previous value of `cur_term`. Otherwise, it returns a null pointer.

On success, the other functions return `OK`. Otherwise, they return `ERR`.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `baudrate(3XCURSES)`, `beep(3XCURSES)`, `initscr(3XCURSES)`, `libcurses(3XCURSES)`, `mvcur(3XCURSES)`, `tigetflag(3XCURSES)`, `use_env(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | form_field_attributes, set_field_fore, field_fore, set_field_back, field_back, set_field_pad, field_pad – format the general display attributes of forms | | | | | | |
|----------------------|--|----------------|-------------------------------------|----------------|---------------|----------------|---------------------------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_field_fore(FIELD *field, chtype attr); chtype field_fore(FIELD *field); int set_field_back(FIELD *field, chtype attr); chtype field_back(FIELD *field); int set_field_pad(FIELD *field, int pad); int field_pad(FIELD *field);</pre> | | | | | | |
| DESCRIPTION | <p>set_field_fore() sets the foreground attribute of <i>field</i>. The foreground attribute is the low-level curses display attribute used to display the field contents. field_fore() returns the foreground attribute of <i>field</i>.</p> <p>set_field_back() sets the background attribute of <i>field</i>. The background attribute is the low-level curses display attribute used to display the extent of the field. field_back() returns the background attribute of <i>field</i>.</p> <p>set_field_pad() sets the pad character of <i>field</i> to <i>pad</i>. The pad character is the character used to fill within the field. field_pad() returns the pad character of <i>field</i>.</p> | | | | | | |
| RETURN VALUES | <p>field_fore(), field_back(), and field_pad() return default values if <i>field</i> is NULL. If <i>field</i> is not NULL and is not a valid FIELD pointer, the return value from these routines is undefined.</p> <p>set_field_fore(), set_field_back(), and set_field_pad() return one of the following:</p> <table border="0"> <tr> <td>E_OK</td> <td>The function returned successfully.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> <tr> <td>E_BAD_ARGUMENT</td> <td>An argument is incorrect.</td> </tr> </table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An argument is incorrect. |
| E_OK | The function returned successfully. | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | |
| E_BAD_ARGUMENT | An argument is incorrect. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | | | |

set_field_buffer(3CURSES)

| NAME | form_field_buffer, set_field_buffer, field_buffer, set_field_status, field_status, set_max_field – set and get forms field attributes | | | | | | |
|----------------------|---|----------------|-------------------------------------|----------------|--------------|----------------|---------------------------|
| SYNOPSIS | <pre>cc [<i>flag...</i>] <i>file...</i> -lform -lcurses [<i>library...</i>] #include <form.h> int set_field_buffer(FIELD *field, int buf, char *value); char *field_buffer(FIELD *field, int buf); int set_field_status(FIELD *field, int status); int field_status(FIELD *field); int set_max_field(FIELD *field, int max);</pre> | | | | | | |
| DESCRIPTION | <p>set_field_buffer() sets buffer <i>buf</i> of <i>field</i> to <i>value</i>. Buffer 0 stores the displayed contents of the field. Buffers other than 0 are application specific and not used by the forms library routines. field_buffer() returns the value of <i>field</i> buffer <i>buf</i>.</p> <p>Every field has an associated status flag that is set whenever the contents of field buffer 0 changes. set_field_status() sets the status flag of <i>field</i> to <i>status</i>. field_status() returns the status of <i>field</i>.</p> <p>set_max_field() sets a maximum growth on a dynamic field, or if <i>max</i>=0 turns off any maximum growth.</p> | | | | | | |
| RETURN VALUES | <p>field_buffer() returns NULL on error.</p> <p>field_status() returns TRUE or FALSE.</p> <p>set_field_buffer(), set_field_status(), and set_max_field() return one of the following:</p> <table><tr><td>E_OK</td><td>The function returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error</td></tr><tr><td>E_BAD_ARGUMENT</td><td>An argument is incorrect.</td></tr></table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error | E_BAD_ARGUMENT | An argument is incorrect. |
| E_OK | The function returned successfully. | | | | | | |
| E_SYSTEM_ERROR | System error | | | | | | |
| E_BAD_ARGUMENT | An argument is incorrect. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | | | |

| NAME | form_field_attributes, set_field_fore, field_fore, set_field_back, field_back, set_field_pad, field_pad – format the general display attributes of forms | | | | | | |
|----------------------|--|----------------|-------------------------------------|----------------|---------------|----------------|---------------------------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_field_fore(FIELD *field, chtype attr); chtype field_fore(FIELD *field); int set_field_back(FIELD *field, chtype attr); chtype field_back(FIELD *field); int set_field_pad(FIELD *field, int pad); int field_pad(FIELD *field);</pre> | | | | | | |
| DESCRIPTION | <p>set_field_fore() sets the foreground attribute of <i>field</i>. The foreground attribute is the low-level curses display attribute used to display the field contents. field_fore() returns the foreground attribute of <i>field</i>.</p> <p>set_field_back() sets the background attribute of <i>field</i>. The background attribute is the low-level curses display attribute used to display the extent of the field. field_back() returns the background attribute of <i>field</i>.</p> <p>set_field_pad() sets the pad character of <i>field</i> to <i>pad</i>. The pad character is the character used to fill within the field. field_pad() returns the pad character of <i>field</i>.</p> | | | | | | |
| RETURN VALUES | <p>field_fore(), field_back(), and field_pad() return default values if <i>field</i> is NULL. If <i>field</i> is not NULL and is not a valid FIELD pointer, the return value from these routines is undefined.</p> <p>set_field_fore(), set_field_back(), and set_field_pad() return one of the following:</p> <table border="0"> <tr> <td style="padding-right: 20px;">E_OK</td> <td>The function returned successfully.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> <tr> <td>E_BAD_ARGUMENT</td> <td>An argument is incorrect.</td> </tr> </table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An argument is incorrect. |
| E_OK | The function returned successfully. | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | |
| E_BAD_ARGUMENT | An argument is incorrect. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | | | |

set_field_init(3CURSES)

| | | | | | |
|----------------------|---|------|-------------------------------------|----------------|---------------|
| NAME | form_hook, set_form_init, form_init, set_form_term, form_term, set_field_init, field_init, set_field_term, field_term – assign application-specific routines for invocation by forms | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_form_init(FORM *form, void (*func)(FORM*)); void (*form_init)(FORM *form); int set_form_term(FORM *form, void (*func)(FORM*)); void (*form_term)(FORM *form); int set_field_init(FORM *form, void (*func)(FORM*)); void (*field_init)(FORM *form); int set_field_term(FORM *form, void (*func)(FORM*)); void (*field_term)(FORM *form);</pre> | | | | |
| DESCRIPTION | <p>These routines allow the programmer to assign application specific routines to be executed automatically at initialization and termination points in the forms application. The user need not specify any application-defined initialization or termination routines at all, but they may be helpful for displaying messages or page numbers and other chores.</p> <p>set_form_init() assigns an application-defined initialization function to be called when the <i>form</i> is posted and just after a page change. form_init() returns a pointer to the initialization function, if any.</p> <p>set_form_term() assigns an application-defined function to be called when the <i>form</i> is unposted and just before a page change. form_term() returns a pointer to the function, if any.</p> <p>set_field_init() assigns an application-defined function to be called when the <i>form</i> is posted and just after the current field changes. field_init() returns a pointer to the function, if any.</p> <p>set_field_term() assigns an application-defined function to be called when the <i>form</i> is unposted and just before the current field changes. field_term() returns a pointer to the function, if any.</p> | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <table><tr><td>E_OK</td><td>The function returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr></table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. |
| E_OK | The function returned successfully. | | | | |
| E_SYSTEM_ERROR | System error. | | | | |

set_field_init(3CURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3CURSES)`, `forms(3CURSES)`, `attributes(5)`

NOTES The header `<form.h>` automatically includes the headers `<eti.h>` and `<curses.h>`.

set_field_just(3CURSES)

| NAME | form_field_just, set_field_just, field_just – format the general appearance of forms | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_field_just(FIELD *field, int justification); int field_just(FIELD *field);</pre> | | | | |
| DESCRIPTION | <p>set_field_just() sets the justification for <i>field</i>. Justification may be one of:</p> <p>NO_JUSTIFICATION JUSTIFY_RIGHT JUSTIFY_LEFT JUSTIFY_CENTER</p> <p>.</p> <p>The field justification will be ignored if <i>field</i> is a dynamic field.</p> <p>field_just() returns the type of justification assigned to <i>field</i>.</p> | | | | |
| RETURN VALUES | <p>field_just() returns one of the following:</p> <p>NO_JUSTIFICATION JUSTIFY_RIGHT JUSTIFY_LEFT JUSTIFY_CENTER.</p> <p>set_field_just() returns one of the following:</p> <p>E_OK The function returned successfully. E_SYSTEM_ERROR System error. E_BAD_ARGUMENT An argument is incorrect.</p> | | | | |
| ATTRIBUTES | <p>See attributes(5) for descriptions of the following attributes:</p> <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | |

set_field_opts(3CURSES)

| | | | | | | | | | | | | | | | | | | | | | |
|----------------------|---|-----------|-------------------------------------|----------------|---|-----------|--|--------|--------------------------|--------|---|---------|---|------------|---|----------|------------------------------------|----------|--------------------------------------|----------|--|
| NAME | form_field_opts, set_field_opts, field_opts_on, field_opts_off, field_opts – forms field option routines | | | | | | | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_field_opts(FIELD *field, OPTIONS opts); int set_field_opts(FIELD *field, OPTIONS opts); int field_opts_on(FIELD *field, OPTIONS opts); int field_opts_off(FIELD *field, OPTIONS opts); OPTIONS field_opts(FIELD *field);</pre> | | | | | | | | | | | | | | | | | | | | |
| DESCRIPTION | <p>set_field_opts() turns on the named options of <i>field</i> and turns off all remaining options. Options are boolean values that can be OR-ed together.</p> <p>field_opts_on() turns on the named options; no other options are changed.</p> <p>field_opts_off() turns off the named options; no other options are changed.</p> <p>field_opts() returns the options set for <i>field</i>.</p> <table><tr><td>O_VISIBLE</td><td>The field is displayed.</td></tr><tr><td>O_ACTIVE</td><td>The field is visited during processing.</td></tr><tr><td>O_PUBLIC</td><td>The field contents are displayed as data is entered.</td></tr><tr><td>O_EDIT</td><td>The field can be edited.</td></tr><tr><td>O_WRAP</td><td>Words not fitting on a line are wrapped to the next line.</td></tr><tr><td>O_BLANK</td><td>The whole field is cleared if a character is entered in the first position.</td></tr><tr><td>O_AUTOSKIP</td><td>Skip to the next field when the current field becomes full.</td></tr><tr><td>O_NULLOK</td><td>A blank field is considered valid.</td></tr><tr><td>O_STATIC</td><td>The field buffers are fixed in size.</td></tr><tr><td>O_PASSOK</td><td>Validate field only if modified by user.</td></tr></table> | O_VISIBLE | The field is displayed. | O_ACTIVE | The field is visited during processing. | O_PUBLIC | The field contents are displayed as data is entered. | O_EDIT | The field can be edited. | O_WRAP | Words not fitting on a line are wrapped to the next line. | O_BLANK | The whole field is cleared if a character is entered in the first position. | O_AUTOSKIP | Skip to the next field when the current field becomes full. | O_NULLOK | A blank field is considered valid. | O_STATIC | The field buffers are fixed in size. | O_PASSOK | Validate field only if modified by user. |
| O_VISIBLE | The field is displayed. | | | | | | | | | | | | | | | | | | | | |
| O_ACTIVE | The field is visited during processing. | | | | | | | | | | | | | | | | | | | | |
| O_PUBLIC | The field contents are displayed as data is entered. | | | | | | | | | | | | | | | | | | | | |
| O_EDIT | The field can be edited. | | | | | | | | | | | | | | | | | | | | |
| O_WRAP | Words not fitting on a line are wrapped to the next line. | | | | | | | | | | | | | | | | | | | | |
| O_BLANK | The whole field is cleared if a character is entered in the first position. | | | | | | | | | | | | | | | | | | | | |
| O_AUTOSKIP | Skip to the next field when the current field becomes full. | | | | | | | | | | | | | | | | | | | | |
| O_NULLOK | A blank field is considered valid. | | | | | | | | | | | | | | | | | | | | |
| O_STATIC | The field buffers are fixed in size. | | | | | | | | | | | | | | | | | | | | |
| O_PASSOK | Validate field only if modified by user. | | | | | | | | | | | | | | | | | | | | |
| RETURN VALUES | <p>set_field_opts, field_opts_on and field_opts_off return one of the following:</p> <table><tr><td>E_OK</td><td>The function returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr><tr><td>E_CURRENT</td><td>The field is the current field.</td></tr></table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. | E_CURRENT | The field is the current field. | | | | | | | | | | | | | | |
| E_OK | The function returned successfully. | | | | | | | | | | | | | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | | | | | | | | | | | | | |
| E_CURRENT | The field is the current field. | | | | | | | | | | | | | | | | | | | | |

set_field_opts(3CURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `courses(3CURSES)`, `forms(3CURSES)`, `attributes(5)`

NOTES The header `<form.h>` automatically includes the headers `<eti.h>` and `<courses.h>`.

| NAME | form_field_attributes, set_field_fore, field_fore, set_field_back, field_back, set_field_pad, field_pad – format the general display attributes of forms | | | | | | |
|----------------------|--|----------------|-------------------------------------|----------------|---------------|----------------|---------------------------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_field_fore(FIELD *field, chtype attr); chtype field_fore(FIELD *field); int set_field_back(FIELD *field, chtype attr); chtype field_back(FIELD *field); int set_field_pad(FIELD *field, int pad); int field_pad(FIELD *field);</pre> | | | | | | |
| DESCRIPTION | <p>set_field_fore() sets the foreground attribute of <i>field</i>. The foreground attribute is the low-level curses display attribute used to display the field contents. field_fore() returns the foreground attribute of <i>field</i>.</p> <p>set_field_back() sets the background attribute of <i>field</i>. The background attribute is the low-level curses display attribute used to display the extent of the field. field_back() returns the background attribute of <i>field</i>.</p> <p>set_field_pad() sets the pad character of <i>field</i> to <i>pad</i>. The pad character is the character used to fill within the field. field_pad() returns the pad character of <i>field</i>.</p> | | | | | | |
| RETURN VALUES | <p>field_fore(), field_back(), and field_pad() return default values if <i>field</i> is NULL. If <i>field</i> is not NULL and is not a valid FIELD pointer, the return value from these routines is undefined.</p> <p>set_field_fore(), set_field_back(), and set_field_pad() return one of the following:</p> <table border="0"> <tr> <td style="padding-right: 20px;">E_OK</td> <td>The function returned successfully.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> <tr> <td>E_BAD_ARGUMENT</td> <td>An argument is incorrect.</td> </tr> </table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An argument is incorrect. |
| E_OK | The function returned successfully. | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | |
| E_BAD_ARGUMENT | An argument is incorrect. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | | | |

set_field_status(3CURSES)

| NAME | form_field_buffer, set_field_buffer, field_buffer, set_field_status, field_status, set_max_field – set and get forms field attributes | | | | | | |
|----------------------|---|----------------|-------------------------------------|----------------|--------------|----------------|---------------------------|
| SYNOPSIS | <pre>cc [<i>flag...</i>] <i>file...</i> -lform -lcurses [<i>library...</i>] #include <form.h> int set_field_buffer(FIELD *field, int buf, char *value); char *field_buffer(FIELD *field, int buf); int set_field_status(FIELD *field, int status); int field_status(FIELD *field); int set_max_field(FIELD *field, int max);</pre> | | | | | | |
| DESCRIPTION | <p>set_field_buffer() sets buffer <i>buf</i> of <i>field</i> to <i>value</i>. Buffer 0 stores the displayed contents of the field. Buffers other than 0 are application specific and not used by the forms library routines. field_buffer() returns the value of <i>field</i> buffer <i>buf</i>.</p> <p>Every field has an associated status flag that is set whenever the contents of field buffer 0 changes. set_field_status() sets the status flag of <i>field</i> to <i>status</i>. field_status() returns the status of <i>field</i>.</p> <p>set_max_field() sets a maximum growth on a dynamic field, or if <i>max</i>=0 turns off any maximum growth.</p> | | | | | | |
| RETURN VALUES | <p>field_buffer() returns NULL on error.</p> <p>field_status() returns TRUE or FALSE.</p> <p>set_field_buffer(), set_field_status(), and set_max_field() return one of the following:</p> <table><tr><td>E_OK</td><td>The function returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error</td></tr><tr><td>E_BAD_ARGUMENT</td><td>An argument is incorrect.</td></tr></table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error | E_BAD_ARGUMENT | An argument is incorrect. |
| E_OK | The function returned successfully. | | | | | | |
| E_SYSTEM_ERROR | System error | | | | | | |
| E_BAD_ARGUMENT | An argument is incorrect. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | | | |

set_field_term(3CURSES)

| | | | | | |
|----------------------|---|------|-------------------------------------|----------------|---------------|
| NAME | form_hook, set_form_init, form_init, set_form_term, form_term, set_field_init, field_init, set_field_term, field_term – assign application-specific routines for invocation by forms | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_form_init(FORM *form, void (*func) (FORM*)); void (*form_init) (FORM *form); int set_form_term(FORM *form, void (*func) (FORM*)); void (*form_term) (FORM *form); int set_field_init(FORM *form, void (*func) (FORM*)); void (*field_init) (FORM *form); int set_field_term(FORM *form, void (*func) (FORM*)); void (*field_term) (FORM *form);</pre> | | | | |
| DESCRIPTION | <p>These routines allow the programmer to assign application specific routines to be executed automatically at initialization and termination points in the forms application. The user need not specify any application-defined initialization or termination routines at all, but they may be helpful for displaying messages or page numbers and other chores.</p> <p>set_form_init() assigns an application-defined initialization function to be called when the form is posted and just after a page change. form_init() returns a pointer to the initialization function, if any.</p> <p>set_form_term() assigns an application-defined function to be called when the form is unposted and just before a page change. form_term() returns a pointer to the function, if any.</p> <p>set_field_init() assigns an application-defined function to be called when the form is posted and just after the current field changes. field_init() returns a pointer to the function, if any.</p> <p>set_field_term() assigns an application-defined function to be called when the form is unposted and just before the current field changes. field_term() returns a pointer to the function, if any.</p> | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <table><tr><td>E_OK</td><td>The function returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr></table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. |
| E_OK | The function returned successfully. | | | | |
| E_SYSTEM_ERROR | System error. | | | | |

set_field_term(3CURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `courses(3CURSES)`, `forms(3CURSES)`, `attributes(5)`

NOTES The header `<form.h>` automatically includes the headers `<eti.h>` and `<courses.h>`.

| NAME | form_field_validation, set_field_type, field_type, field_arg – forms field data type validation | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_field_type(FIELD *field, FIELDTYPE *type, ...); FIELDTYPE *field_type(FIELD *field); char *field_arg(FIELD *field);</pre> | | | | |
| DESCRIPTION | <p>set_field_type() associates the specified field type with <i>field</i>. Certain field types take additional arguments. TYPE_ALNUM, for instance, requires one, the minimum width specification for the field. The other predefined field types are: TYPE_ALPHA, TYPE_ENUM, TYPE_INTEGER, TYPE_NUMERIC, and TYPE_REGEX.</p> <p>field_type() returns a pointer to the field type of <i>field</i>. NULL is returned if no field type is assigned.</p> <p>field_arg() returns a pointer to the field arguments associated with the field type of <i>field</i>. NULL is returned if no field type is assigned.</p> | | | | |
| RETURN VALUES | <p>field_type() and field_arg() return NULL on error.</p> <p>set_field_type() returns one of the following:</p> <p>E_OK The function returned successfully.</p> <p>E_SYSTEM_ERROR System error.</p> | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | |

set_fieldtype_arg(3CURSES)

| | | | | | | | | | |
|----------------------|--|------|-------------------------------------|----------------|---------------|----------------|---------------------------|-------------|--|
| NAME | form_fieldtype, new_fieldtype, free_fieldtype, set_fieldtype_arg, set_fieldtype_choice, link_fieldtype – forms fieldtype routines | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> FIELDTYPE *new_fieldtype(int (* field_check) (FIELD *, char *), int (*char_check) (int, char *)); int free_fieldtype(FIELDTYPE *fieldtype); int set_fieldtype_arg(FIELDTYPE *fieldtype, char *(* mak_arg) (va_list *), char *(* copy_arg) (char *), void (* free_arg) (char *)); int set_fieldtype_choice(FIELDTYPE *fieldtype, int (* next_choice) (FIELD *, char *), int (*prev_choice) (FIELD *, char *)); FIELDTYPE *link_fieldtype(FIELDTYPE *type1, FIELDTYPE *type2);</pre> | | | | | | | | |
| DESCRIPTION | <p><code>new_fieldtype()</code> creates a new field type. The application programmer must write the function <code>field_check</code>, which validates the field value, and the function <code>char_check</code>, which validates each character. <code>free_fieldtype()</code> frees the space allocated for the field type.</p> <p>By associating function pointers with a field type, <code>set_fieldtype_arg()</code> connects to the field type additional arguments necessary for a <code>set_field_type()</code> call. Function <code>mak_arg</code> allocates a structure for the field specific parameters to <code>set_field_type()</code> and returns a pointer to the saved data. Function <code>copy_arg</code> duplicates the structure created by <code>make_arg</code>. Function <code>free_arg</code> frees any storage allocated by <code>make_arg</code> or <code>copy_arg</code>.</p> <p>The <code>form_driver()</code> requests <code>REQ_NEXT_CHOICE</code> and <code>REQ_PREV_CHOICE</code> let the user request the next or previous value of a field type comprising an ordered set of values. <code>set_fieldtype_choice()</code> allows the application programmer to implement these requests for the given field type. It associates with the given field type those application-defined functions that return pointers to the next or previous choice for the field.</p> <p><code>link_fieldtype()</code> returns a pointer to the field type built from the two given types. The constituent types may be any application-defined or pre-defined types.</p> | | | | | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <table><tr><td>E_OK</td><td>The function returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr><tr><td>E_BAD_ARGUMENT</td><td>An argument is incorrect.</td></tr><tr><td>E_CONNECTED</td><td>Type is connected to one or more fields.</td></tr></table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An argument is incorrect. | E_CONNECTED | Type is connected to one or more fields. |
| E_OK | The function returned successfully. | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | |
| E_BAD_ARGUMENT | An argument is incorrect. | | | | | | | | |
| E_CONNECTED | Type is connected to one or more fields. | | | | | | | | |

set_fieldtype_arg(3CURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3CURSES)`, `forms(3CURSES)`, `attributes(5)`

NOTES The header `<form.h>` automatically includes the headers `<eti.h>` and `< curses.h>`.

set_fieldtype_choice(3CURSES)

| | | | | | | | | | |
|----------------------|--|------|-------------------------------------|----------------|---------------|----------------|---------------------------|-------------|--|
| NAME | form_fieldtype, new_fieldtype, free_fieldtype, set_fieldtype_arg, set_fieldtype_choice, link_fieldtype – forms fieldtype routines | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> FIELDTYPE *new_fieldtype(int (* field_check)(FIELD *, char *), int (*char_check)(int, char *)); int free_fieldtype(FIELDTYPE *fieldtype); int set_fieldtype_arg(FIELDTYPE *fieldtype, char *(* mak_arg)(va_list *), char *(* copy_arg)(char *), void (* free_arg)(char *)); int set_fieldtype_choice(FIELDTYPE *fieldtype, int (* next_choice)(FIELD *, char *), int (*prev_choice)(FIELD *, char *)); FIELDTYPE *link_fieldtype(FIELDTYPE *type1, FIELDTYPE *type2);</pre> | | | | | | | | |
| DESCRIPTION | <p>new_fieldtype() creates a new field type. The application programmer must write the function <i>field_check</i>, which validates the field value, and the function <i>char_check</i>, which validates each character. free_fieldtype() frees the space allocated for the field type.</p> <p>By associating function pointers with a field type, set_fieldtype_arg() connects to the field type additional arguments necessary for a set_field_type() call. Function <i>mak_arg</i> allocates a structure for the field specific parameters to set_field_type() and returns a pointer to the saved data. Function <i>copy_arg</i> duplicates the structure created by <i>make_arg</i>. Function <i>free_arg</i> frees any storage allocated by <i>make_arg</i> or <i>copy_arg</i>.</p> <p>The form_driver() requests REQ_NEXT_CHOICE and REQ_PREV_CHOICE let the user request the next or previous value of a field type comprising an ordered set of values. set_fieldtype_choice() allows the application programmer to implement these requests for the given field type. It associates with the given field type those application-defined functions that return pointers to the next or previous choice for the field.</p> <p>link_fieldtype() returns a pointer to the field type built from the two given types. The constituent types may be any application-defined or pre-defined types.</p> | | | | | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <table><tr><td>E_OK</td><td>The function returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr><tr><td>E_BAD_ARGUMENT</td><td>An argument is incorrect.</td></tr><tr><td>E_CONNECTED</td><td>Type is connected to one or more fields.</td></tr></table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An argument is incorrect. | E_CONNECTED | Type is connected to one or more fields. |
| E_OK | The function returned successfully. | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | |
| E_BAD_ARGUMENT | An argument is incorrect. | | | | | | | | |
| E_CONNECTED | Type is connected to one or more fields. | | | | | | | | |

set_fieldtype_choice(3CURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `courses(3CURSES)`, `forms(3CURSES)`, `attributes(5)`

NOTES The header `<form.h>` automatically includes the headers `<eti.h>` and `<courses.h>`.

set_field_userptr(3CURSES)

NAME | form_field_userptr, set_field_userptr, field_userptr – associate application data with forms

SYNOPSIS |

```
cc [ flag... ] file... -lform -lcurses [ library... ]
#include <form.h>

int set_field_userptr(FIELD *field, char *ptr);
char *field_userptr(FIELD *field);
```

DESCRIPTION | Every field has an associated user pointer that can be used to store pertinent data. set_field_userptr() sets the user pointer of *field*. field_userptr() returns the user pointer of *field*.

RETURN VALUES | field_userptr() returns NULL on error. set_field_userptr() returns one of the following:

E_OK | The function returned successfully.

E_SYSTEM_ERROR | System error.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5)

NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>.

NAME form_field, set_form_fields, form_fields, field_count, move_field – connect fields to forms

SYNOPSIS

```
cc [ flag... ] file... -lform -lcurses [ library... ]
#include <form.h>

int set_form_fields(FORM *form, FIELD **field);
FIELD **form_fields(FORM *form);
int field_count(FORM *form);
int move_field(FIELD *field, int frow, int fcol);
```

DESCRIPTION

set_form_fields() changes the fields connected to *form* to *fields*. The original fields are disconnected.

form_fields() returns a pointer to the field pointer array connected to *form*.

field_count() returns the number of fields connected to *form*.

move_field() moves the disconnected *field* to the location *frow*, *fcol* in the forms subwindow.

RETURN VALUES

form_fields() returns NULL on error.

field_count() returns -1 on error.

set_form_fields() and move_field() return one of the following:

| | |
|----------------|---|
| E_OK | The function returned successfully. |
| E_CONNECTED | The field is already connected to a form. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An argument is incorrect |
| E_POSTED | The form is posted. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), forms(3CURSES), attributes(5)

NOTES The header <form.h> automatically includes the headers <eti.h> and <curses.h>.

set_form_init(3CURSES)

| | | | | | |
|----------------------|---|------|-------------------------------------|----------------|---------------|
| NAME | form_hook, set_form_init, form_init, set_form_term, form_term, set_field_init, field_init, set_field_term, field_term – assign application-specific routines for invocation by forms | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_form_init(FORM *form, void (*func)(FORM*)); void (*form_init)(FORM *form); int set_form_term(FORM *form, void (*func)(FORM*)); void (*form_term)(FORM *form); int set_field_init(FORM *form, void (*func)(FORM*)); void (*field_init)(FORM *form); int set_field_term(FORM *form, void (*func)(FORM*)); void (*field_term)(FORM *form);</pre> | | | | |
| DESCRIPTION | <p>These routines allow the programmer to assign application specific routines to be executed automatically at initialization and termination points in the forms application. The user need not specify any application-defined initialization or termination routines at all, but they may be helpful for displaying messages or page numbers and other chores.</p> <p>set_form_init() assigns an application-defined initialization function to be called when the <i>form</i> is posted and just after a page change. form_init() returns a pointer to the initialization function, if any.</p> <p>set_form_term() assigns an application-defined function to be called when the <i>form</i> is unposted and just before a page change. form_term() returns a pointer to the function, if any.</p> <p>set_field_init() assigns an application-defined function to be called when the <i>form</i> is posted and just after the current field changes. field_init() returns a pointer to the function, if any.</p> <p>set_field_term() assigns an application-defined function to be called when the <i>form</i> is unposted and just before the current field changes. field_term() returns a pointer to the function, if any.</p> | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <table><tr><td>E_OK</td><td>The function returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr></table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. |
| E_OK | The function returned successfully. | | | | |
| E_SYSTEM_ERROR | System error. | | | | |

set_form_init(3CURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3CURSES)`, `forms(3CURSES)`, `attributes(5)`

NOTES The header `<form.h>` automatically includes the headers `<eti.h>` and `< curses.h>`.

set_form_opts(3CURSES)

| NAME | form_opts, set_form_opts, form_opts_on, form_opts_off – forms option routines | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_form_opts(FORM *form, OPTIONS opts); int form_opts_on(FORM *form, OPTIONS opts); int form_opts_off(FORM *form, OPTIONS opts); OPTIONS form_opts(FORM *form);</pre> | | | | |
| DESCRIPTION | <p>set_form_opts() turns on the named options for <i>form</i> and turns off all remaining options. Options are boolean values which can be OR-ed together. form_opts_on() turns on the named options; no other options are changed. form_opts_off() turns off the named options; no other options are changed.</p> <p>form_opts() returns the options set for <i>form</i>.</p> <p>O_NL_OVERLOAD Overload the REQ_NEW_LINE form driver request.</p> <p>O_BS_OVERLOAD Overload the REQ_DEL_PREV form driver request.</p> | | | | |
| RETURN VALUES | <p>set_form_opts(), form_opts_on(), and form_opts_off() return one of the following:</p> <p>E_OK The function returned successfully.</p> <p>E_SYSTEM_ERROR System error.</p> | | | | |
| ATTRIBUTES | See attributes (5) for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | |

NAME form_page, set_form_page, set_current_field, current_field, field_index – set forms current page and field

SYNOPSIS

```
cc [ flag... ] file... -lform -lcurses [ library... ]
#include <form.h>

int set_form_page(FORM *form, int page);
int form_page(FORM *form);
int set_current_field(FORM *form, FIELD *field);
FIELD *current_field(FORM*form);
int field_index(FIELD *field);
```

DESCRIPTION

set_form_page() sets the page number of *form* to *page*. form_page() returns the current page number of *form*.

set_current_field() sets the current field of *form* to *field*. current_field() returns a pointer to the current field of *form*.

field_index() returns the index in the field pointer array of *field*.

RETURN VALUES

form_page() returns -1 on error.

current_field() returns NULL on error.

field_index() returns -1 on error.

set_form_page() and set_current_field() return one of the following:

| | |
|------------------|--|
| E_OK | The function returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An argument is incorrect. |
| E_BAD_STATE | The routine was called from an initialization or termination function. |
| E_INVALID_FIELD | The field contents are invalid. |
| E_REQUEST_DENIED | The form driver request failed |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), forms(3CURSES), attributes(5)

set_form_page(3CURSES)

NOTES | The header `<form.h>` automatically includes the headers `<eti.h>` and `<curses.h>`.

| NAME | form_win, set_form_win, set_form_sub, form_sub, scale_form – forms window and subwindow association routines | | | | | | | | | | |
|----------------------|---|----------------|-------------------------------------|----------------|---------------|----------------|---------------------------|-----------------|---------------------------------------|----------|---------------------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_form_win(FORM *form, WINDOW *win); WINDOW *form_win(FORM *form); int set_form_sub(FORM *form, WINDOW *sub); WINDOW *form_sub(FORM *form); int scale_form(FORM *form, int *rows, int *cols);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>set_form_win() sets the window of <i>form</i> to <i>win</i>. form_win() returns a pointer to the window associated with <i>form</i>. set_form_sub() sets the subwindow of <i>form</i> to <i>sub</i>. form_sub() returns a pointer to the subwindow associated with <i>form</i>. scale_form() returns the smallest window size necessary for the subwindow of <i>form</i>. <i>rows</i> and <i>cols</i> are pointers to the locations used to return the number of rows and columns for the form.</p> | | | | | | | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <table border="0"> <tr> <td>E_OK</td> <td>The function returned successfully.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> <tr> <td>E_BAD_ARGUMENT</td> <td>An argument is incorrect.</td> </tr> <tr> <td>E_NOT_CONNECTED</td> <td>The field is not connected to a form.</td> </tr> <tr> <td>E_POSTED</td> <td>The form is posted.</td> </tr> </table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An argument is incorrect. | E_NOT_CONNECTED | The field is not connected to a form. | E_POSTED | The form is posted. |
| E_OK | The function returned successfully. | | | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | | | |
| E_BAD_ARGUMENT | An argument is incorrect. | | | | | | | | | | |
| E_NOT_CONNECTED | The field is not connected to a form. | | | | | | | | | | |
| E_POSTED | The form is posted. | | | | | | | | | | |
| ATTRIBUTES | <p>See attributes(5) for descriptions of the following attributes:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | | | | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | | | | | |
| MT-Level | Unsafe | | | | | | | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | | | | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | | | | | | | |

set_form_term(3CURSES)

| | | | | | |
|----------------------|---|------|-------------------------------------|----------------|---------------|
| NAME | form_hook, set_form_init, form_init, set_form_term, form_term, set_field_init, field_init, set_field_term, field_term – assign application-specific routines for invocation by forms | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_form_init(FORM *form, void (*func)(FORM*)); void (*form_init)(FORM *form); int set_form_term(FORM *form, void (*func)(FORM*)); void (*form_term)(FORM *form); int set_field_init(FORM *form, void (*func)(FORM*)); void (*field_init)(FORM *form); int set_field_term(FORM *form, void (*func)(FORM*)); void (*field_term)(FORM *form);</pre> | | | | |
| DESCRIPTION | <p>These routines allow the programmer to assign application specific routines to be executed automatically at initialization and termination points in the forms application. The user need not specify any application-defined initialization or termination routines at all, but they may be helpful for displaying messages or page numbers and other chores.</p> <p>set_form_init() assigns an application-defined initialization function to be called when the <i>form</i> is posted and just after a page change. form_init() returns a pointer to the initialization function, if any.</p> <p>set_form_term() assigns an application-defined function to be called when the <i>form</i> is unposted and just before a page change. form_term() returns a pointer to the function, if any.</p> <p>set_field_init() assigns an application-defined function to be called when the <i>form</i> is posted and just after the current field changes. field_init() returns a pointer to the function, if any.</p> <p>set_field_term() assigns an application-defined function to be called when the <i>form</i> is unposted and just before the current field changes. field_term() returns a pointer to the function, if any.</p> | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <table><tr><td>E_OK</td><td>The function returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr></table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. |
| E_OK | The function returned successfully. | | | | |
| E_SYSTEM_ERROR | System error. | | | | |

set_form_term(3CURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3CURSES)`, `forms(3CURSES)`, `attributes(5)`

NOTES The header `<form.h>` automatically includes the headers `<eti.h>` and `< curses.h>`.

set_form_userptr(3CURSES)

| NAME | form_userptr, set_form_userptr – associate application data with forms | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_form_userptr(FORM *form, char *ptr); char *form_userptr(FORM *form);</pre> | | | | |
| DESCRIPTION | Every form has an associated user pointer that can be used to store pertinent data. set_form_userptr() sets the user pointer of <i>form</i> . form_userptr() returns the user pointer of <i>form</i> . | | | | |
| RETURN VALUES | form_userptr() returns NULL on error. set_form_userptr() returns one of the following: E_OK The function returned successfully. E_SYSTEM_ERROR System error. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: <table border="1" data-bbox="444 909 1414 999"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | |

| NAME | form_win, set_form_win, set_form_sub, form_sub, scale_form – forms window and subwindow association routines | | | | | | | | | | |
|----------------------|---|----------------|-------------------------------------|----------------|---------------|----------------|---------------------------|-----------------|---------------------------------------|----------|---------------------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int set_form_win(FORM *form, WINDOW *win); WINDOW *form_win(FORM *form); int set_form_sub(FORM *form, WINDOW *sub); WINDOW *form_sub(FORM *form); int scale_form(FORM *form, int *rows, int *cols);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>set_form_win() sets the window of <i>form</i> to <i>win</i>. form_win() returns a pointer to the window associated with <i>form</i>. set_form_sub() sets the subwindow of <i>form</i> to <i>sub</i>. form_sub() returns a pointer to the subwindow associated with <i>form</i>. scale_form() returns the smallest window size necessary for the subwindow of <i>form</i>. <i>rows</i> and <i>cols</i> are pointers to the locations used to return the number of rows and columns for the form.</p> | | | | | | | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <table border="0"> <tr> <td>E_OK</td> <td>The function returned successfully.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> <tr> <td>E_BAD_ARGUMENT</td> <td>An argument is incorrect.</td> </tr> <tr> <td>E_NOT_CONNECTED</td> <td>The field is not connected to a form.</td> </tr> <tr> <td>E_POSTED</td> <td>The form is posted.</td> </tr> </table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An argument is incorrect. | E_NOT_CONNECTED | The field is not connected to a form. | E_POSTED | The form is posted. |
| E_OK | The function returned successfully. | | | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | | | |
| E_BAD_ARGUMENT | An argument is incorrect. | | | | | | | | | | |
| E_NOT_CONNECTED | The field is not connected to a form. | | | | | | | | | | |
| E_POSTED | The form is posted. | | | | | | | | | | |
| ATTRIBUTES | <p>See attributes(5) for descriptions of the following attributes:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | | | | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | | | | | |
| MT-Level | Unsafe | | | | | | | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5) | | | | | | | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | | | | | | | |

set_item_init(3CURSES)

| | | | | | |
|----------------------|--|------|------------------------------------|----------------|---------------|
| NAME | menu_hook, set_item_init, item_init, set_item_term, item_term, set_menu_init, menu_init, set_menu_term, menu_term – assign application-specific routines for automatic invocation by menus | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int set_item_init(MENU *menu, void (*func)(MENU *)); int set_item_term(MENU *menu, void (*func)(MENU *)); void item_term(MENU *menu); int set_menu_init(MENU *menu, void (*func)(MENU *)); void menu_init(MENU *menu); int set_menu_term(MENU *menu, void (*func)(MENU *)); void menu_term(MENU *menu);</pre> | | | | |
| DESCRIPTION | <p>set_item_init() assigns the application-defined function to be called when the <i>menu</i> is posted and just after the current item changes. item_init() returns a pointer to the item initialization routine, if any, called when the <i>menu</i> is posted and just after the current item changes.</p> <p>set_item_term() assigns an application-defined function to be called when the <i>menu</i> is unposted and just before the current item changes. item_term() returns a pointer to the termination function, if any, called when the <i>menu</i> is unposted and just before the current item changes.</p> <p>set_menu_init() assigns an application-defined function to be called when the <i>menu</i> is posted and just after the top row changes on a posted menu. menu_init() returns a pointer to the menu initialization routine, if any, called when the <i>menu</i> is posted and just after the top row changes on a posted menu.</p> <p>set_menu_term() assigns an application-defined function to be called when the <i>menu</i> is unposted and just before the top row changes on a posted menu. menu_term() returns a pointer to the menu termination routine, if any, called when the <i>menu</i> is unposted and just before the top row changes on a posted menu.</p> | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <table><tr><td>E_OK</td><td>The routine returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr></table> | E_OK | The routine returned successfully. | E_SYSTEM_ERROR | System error. |
| E_OK | The routine returned successfully. | | | | |
| E_SYSTEM_ERROR | System error. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |

set_item_init(3CURSES)

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), attributes(5)

NOTES The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

set_item_opts(3CURSES)

| NAME | menu_item_opts, set_item_opts, item_opts_on, item_opts_off, item_opts – menus item option routines | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int set_item_opts (ITEM *item, OPTIONS opts); int item_opts_on (ITEM *item, OPTIONS opts); int item_opts_off (ITEM *item, OPTIONS opts); OPTIONS item_opts (ITEM *item);</pre> | | | | |
| DESCRIPTION | <p>set_item_opts() turns on the named options for <i>item</i> and turns off all other options. Options are boolean values that can be OR-ed together.</p> <p>item_opts_on() turns on the named options for <i>item</i>; no other option is changed.</p> <p>item_opts_off() turns off the named options for <i>item</i>; no other option is changed.</p> <p>item_opts() returns the current options of <i>item</i>.</p> <p>O_SELECTABLE The item can be selected during menu processing.</p> | | | | |
| RETURN VALUES | <p>Except for item_opts(), these routines return one of the following:</p> <p>E_OK The routine returned successfully.</p> <p>E_SYSTEM_ERROR System error.</p> | | | | |
| ATTRIBUTES | <p>See attributes(5) for descriptions of the following attributes:</p> <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | courses(3CURSES), menus(3CURSES), attributes(5) | | | | |
| NOTES | The header <menu.h> automatically includes the headers <eti.h> and <curses.h>. | | | | |

| | | | | | |
|----------------------|--|------|------------------------------------|----------------|---------------|
| NAME | menu_hook, set_item_init, item_init, set_item_term, item_term, set_menu_init, menu_init, set_menu_term, menu_term – assign application-specific routines for automatic invocation by menus | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int set_item_init(MENU *menu, void (*func)(MENU *)); int set_item_term(MENU *menu, void (*func)(MENU *)); void item_term(MENU *menu); int set_menu_init(MENU *menu, void (*func)(MENU *)); void menu_init(MENU *menu); int set_menu_term(MENU *menu, void (*func)(MENU *)); void menu_term(MENU *menu);</pre> | | | | |
| DESCRIPTION | <p>set_item_init() assigns the application-defined function to be called when the <i>menu</i> is posted and just after the current item changes. item_init() returns a pointer to the item initialization routine, if any, called when the <i>menu</i> is posted and just after the current item changes.</p> <p>set_item_term() assigns an application-defined function to be called when the <i>menu</i> is unposted and just before the current item changes. item_term() returns a pointer to the termination function, if any, called when the <i>menu</i> is unposted and just before the current item changes.</p> <p>set_menu_init() assigns an application-defined function to be called when the <i>menu</i> is posted and just after the top row changes on a posted menu. menu_init() returns a pointer to the menu initialization routine, if any, called when the <i>menu</i> is posted and just after the top row changes on a posted menu.</p> <p>set_menu_term() assigns an application-defined function to be called when the <i>menu</i> is unposted and just before the top row changes on a posted menu. menu_term() returns a pointer to the menu termination routine, if any, called when the <i>menu</i> is unposted and just before the top row changes on a posted menu.</p> | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <table border="0"> <tr> <td style="padding-right: 20px;">E_OK</td> <td>The routine returned successfully.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> </table> | E_OK | The routine returned successfully. | E_SYSTEM_ERROR | System error. |
| E_OK | The routine returned successfully. | | | | |
| E_SYSTEM_ERROR | System error. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |

set_item_term(3CURSES)

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), attributes(5)

NOTES The header `<menu.h>` automatically includes the headers `<eti.h>` and `<curses.h>`.

| NAME | menu_item_userptr, set_item_userptr, item_userptr – associate application data with menus items | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int set_item_userptr(ITEM *item, char *userptr); char *item_userptr(ITEM *item);</pre> | | | | |
| DESCRIPTION | Every item has an associated user pointer that can be used to store relevant information. <code>set_item_userptr()</code> sets the user pointer of <i>item</i> . <code>item_userptr()</code> returns the user pointer of <i>item</i> . | | | | |
| RETURN VALUES | <p><code>item_userptr()</code> returns NULL on error. <code>set_item_userptr()</code> returns one of the following:</p> <p>E_OK The routine returned successfully.</p> <p>E_SYSTEM_ERROR System error.</p> | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>menus(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header <code><menu.h></code> automatically includes the headers <code><eti.h></code> and <code><curses.h></code> . | | | | |

set_item_value(3CURSES)

| NAME | menu_item_value, set_item_value, item_value – set and get menus item values | | | | | | |
|-------------------------------|---|-------------------|------------------------------------|-----------------------------|---------------|-------------------------------|--|
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int set_item_value (ITEM *item, int bool); int item_value (ITEM *item);</pre> | | | | | | |
| DESCRIPTION | <p>Unlike single-valued menus, multi-valued menus enable the end-user to select one or more items from a menu. <code>set_item_value()</code> sets the selected value of the <i>item</i> — TRUE (selected) or FALSE (not selected). <code>set_item_value()</code> may be used only with multi-valued menus. To make a menu multi-valued, use <code>set_menu_opts</code> or <code>menu_opts_off()</code> to turn off the option <code>O_ONEVALUE</code>. (See <code>menu_opts(3CURSES)</code>).</p> <p><code>item_value()</code> returns the select value of <i>item</i>, either TRUE (selected) or FALSE (unselected).</p> | | | | | | |
| RETURN VALUES | <p><code>set_item_value()</code> returns one of the following:</p> <table><tr><td><code>E_OK</code></td><td>The routine returned successfully.</td></tr><tr><td><code>E_SYSTEM_ERROR</code></td><td>System error.</td></tr><tr><td><code>E_REQUEST_DENIED</code></td><td>The menu driver could not process the request.</td></tr></table> | <code>E_OK</code> | The routine returned successfully. | <code>E_SYSTEM_ERROR</code> | System error. | <code>E_REQUEST_DENIED</code> | The menu driver could not process the request. |
| <code>E_OK</code> | The routine returned successfully. | | | | | | |
| <code>E_SYSTEM_ERROR</code> | System error. | | | | | | |
| <code>E_REQUEST_DENIED</code> | The menu driver could not process the request. | | | | | | |
| ATTRIBUTES | <p>See <code>attributes(5)</code> for descriptions of the following attributes:</p> <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>menus(3CURSES)</code> , <code>menu_opts(3CURSES)</code> , <code>attributes(5)</code> | | | | | | |
| NOTES | The header <code><menu.h></code> automatically includes the headers <code><eti.h></code> and <code><curses.h></code> . | | | | | | |

set_max_field(3CURSES)

NAME form_field_buffer, set_field_buffer, field_buffer, set_field_status, field_status, set_max_field – set and get forms field attributes

SYNOPSIS

```
cc [ flag... ] file... -lform -lcurses [ library... ]
#include <form.h>
```

```
int set_field_buffer(FIELD *field, int buf, char *value);
char *field_buffer(FIELD *field, int buf);
int set_field_status(FIELD *field, int status);
int field_status(FIELD *field);
int set_max_field(FIELD *field, int max);
```

DESCRIPTION set_field_buffer() sets buffer *buf* of *field* to *value*. Buffer 0 stores the displayed contents of the field. Buffers other than 0 are application specific and not used by the forms library routines. field_buffer() returns the value of *field* buffer *buf*.

Every field has an associated status flag that is set whenever the contents of field buffer 0 changes. set_field_status() sets the status flag of *field* to *status*. field_status() returns the status of *field*.

set_max_field() sets a maximum growth on a dynamic field, or if *max*=0 turns off any maximum growth.

RETURN VALUES field_buffer() returns NULL on error.

field_status() returns TRUE or FALSE.

set_field_buffer(), set_field_status(), and set_max_field() return one of the following:

| | |
|----------------|-------------------------------------|
| E_OK | The function returned successfully. |
| E_SYSTEM_ERROR | System error |
| E_BAD_ARGUMENT | An argument is incorrect. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), forms(3CURSES), attributes(5)

NOTES The header <form.h> automatically includes the headers <eti.h> and <curses.h>.

set_menu_back(3CURSES)

NAME menu_attributes, set_menu_fore, menu_fore, set_menu_back, menu_back, set_menu_grey, menu_grey, set_menu_pad, menu_pad – control menus display attributes

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

int set_menu_fore(MENU *menu, chtype attr);
chtype menu_fore(MENU *menu);

int set_menu_back(MENU *menu, chtype attr);
chtype menu_back(MENU *menu);

int set_menu_grey(MENU *menu, chtype attr);
chtype menu_grey(MENU *menu);

int set_menu_pad(MENU *menu, int pad);
int menu_pad(MENU *menu);
```

DESCRIPTION

set_menu_fore() sets the foreground attribute of *menu* — the display attribute for the current item (if selectable) on single-valued menus and for selected items on multi-valued menus. This display attribute is a curses library visual attribute. menu_fore() returns the foreground attribute of *menu*.

set_menu_back() sets the background attribute of *menu* — the display attribute for unselected, yet selectable, items. This display attribute is a curses library visual attribute.

set_menu_grey() sets the grey attribute of *menu* — the display attribute for nonselectable items in multi-valued menus. This display attribute is a curses library visual attribute. menu_grey() returns the grey attribute of *menu*.

The pad character is the character that fills the space between the name and description of an item. set_menu_pad() sets the pad character for *menu* to *pad*. menu_pad() returns the pad character of *menu*.

RETURN VALUES These routines return one of the following:

| | |
|----------------|--|
| E_OK | The routine returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

set_menu_back(3CURSES)

SEE ALSO | curses(3CURSES), menus(3CURSES), attributes(5)

NOTES | The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

set_menu_fore(3CURSES)

SEE ALSO | curses(3CURSES), menus(3CURSES), attributes(5)

NOTES | The header `<menu.h>` automatically includes the headers `<eti.h>` and `<curses.h>`.

set_menu_format(3CURSES)

NAME menu_format, set_menu_format – set and get maximum numbers of rows and columns in menus

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

int set_menu_format(MENU *menu, int rows, int cols);
void menu_format(MENU *menu, int *rows, int *cols);
```

DESCRIPTION set_menu_format() sets the maximum number of rows and columns of items that may be displayed at one time on a menu. If the menu contains more items than can be displayed at once, the menu will be scrollable.

menu_format() returns the maximum number of rows and columns that may be displayed at one time on menu. rows and cols are pointers to the variables used to return these values.

RETURN VALUES set_menu_format() returns one of the following:

| | |
|----------------|--|
| E_OK | The routine returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. |
| E_POSTED | The menu is already posted. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), attributes(5)

NOTES The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

| | | | | | | | |
|----------------------|--|------|------------------------------------|----------------|---------------|----------------|--|
| NAME | menu_attributes, set_menu_fore, menu_fore, set_menu_back, menu_back, set_menu_grey, menu_grey, set_menu_pad, menu_pad – control menus display attributes | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int set_menu_fore(MENU *menu, chtype attr); chtype menu_fore(MENU *menu); int set_menu_back(MENU *menu, chtype attr); chtype menu_back(MENU *menu); int set_menu_grey(MENU*menu, chtype attr); chtype menu_grey(MENU *menu); int set_menu_pad(MENU *menu, int pad); int menu_pad(MENU *menu);</pre> | | | | | | |
| DESCRIPTION | <p>set_menu_fore() sets the foreground attribute of <i>menu</i> — the display attribute for the current item (if selectable) on single-valued menus and for selected items on multi-valued menus. This display attribute is a curses library visual attribute. menu_fore() returns the foreground attribute of <i>menu</i>.</p> <p>set_menu_back() sets the background attribute of <i>menu</i> — the display attribute for unselected, yet selectable, items. This display attribute is a curses library visual attribute.</p> <p>set_menu_grey() sets the grey attribute of <i>menu</i> — the display attribute for nonselectable items in multi-valued menus. This display attribute is a curses library visual attribute. menu_grey() returns the grey attribute of <i>menu</i>.</p> <p>The pad character is the character that fills the space between the name and description of an item. set_menu_pad() sets the pad character for <i>menu</i> to <i>pad</i>. menu_pad() returns the pad character of <i>menu</i>.</p> | | | | | | |
| RETURN VALUES | <p>These routines return one of the following:</p> <table border="0"> <tr> <td style="padding-right: 20px;">E_OK</td> <td>The routine returned successfully.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> <tr> <td>E_BAD_ARGUMENT</td> <td>An incorrect argument was passed to the routine.</td> </tr> </table> | E_OK | The routine returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An incorrect argument was passed to the routine. |
| E_OK | The routine returned successfully. | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

set_menu_grey(3CURSES)

SEE ALSO | curses(3CURSES), menus(3CURSES), attributes(5)

NOTES | The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

| | | | | | |
|----------------------|--|------|------------------------------------|----------------|---------------|
| NAME | menu_hook, set_item_init, item_init, set_item_term, item_term, set_menu_init, menu_init, set_menu_term, menu_term – assign application-specific routines for automatic invocation by menus | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int set_item_init(MENU *menu, void (*func)(MENU *)); int set_item_term(MENU *menu, void (*func)(MENU *)); void item_term(MENU *menu); int set_menu_init(MENU *menu, void (*func)(MENU *)); void menu_init(MENU *menu); int set_menu_term(MENU *menu, void (*func)(MENU *)); void menu_term(MENU *menu);</pre> | | | | |
| DESCRIPTION | <p>set_item_init() assigns the application-defined function to be called when the <i>menu</i> is posted and just after the current item changes. item_init() returns a pointer to the item initialization routine, if any, called when the <i>menu</i> is posted and just after the current item changes.</p> <p>set_item_term() assigns an application-defined function to be called when the <i>menu</i> is unposted and just before the current item changes. item_term() returns a pointer to the termination function, if any, called when the <i>menu</i> is unposted and just before the current item changes.</p> <p>set_menu_init() assigns an application-defined function to be called when the <i>menu</i> is posted and just after the top row changes on a posted menu. menu_init() returns a pointer to the menu initialization routine, if any, called when the <i>menu</i> is posted and just after the top row changes on a posted menu.</p> <p>set_menu_term() assigns an application-defined function to be called when the <i>menu</i> is unposted and just before the top row changes on a posted menu. menu_term() returns a pointer to the menu termination routine, if any, called when the <i>menu</i> is unposted and just before the top row changes on a posted menu.</p> | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <table border="0"> <tr> <td style="padding-right: 20px;">E_OK</td> <td>The routine returned successfully.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> </table> | E_OK | The routine returned successfully. | E_SYSTEM_ERROR | System error. |
| E_OK | The routine returned successfully. | | | | |
| E_SYSTEM_ERROR | System error. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |

set_menu_init(3CURSES)

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), attributes(5)

NOTES The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

NAME menu_items, set_menu_items, item_count – connect and disconnect items to and from menus

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

int set_menu_items(MENU *menu, ITEM **items);
ITEM **menu_items(MENU *menu);
int item_count(MENU *menu);
```

DESCRIPTION set_menu_items() changes the item pointer array connected to *menu* to the item pointer array *items*. menu_items() returns a pointer to the item pointer array connected to *menu*. item_count() returns the number of items in *menu*.

RETURN VALUES menu_items() returns NULL on error.

item_count() returns -1 on error.

set_menu_items() returns one of the following:

| | |
|----------------|--|
| E_OK | The routine returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. |
| E_POSTED | The menu is already posted. |
| E_CONNECTED | One or more items are already connected to another menu. |

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), attributes(5)

NOTES The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

set_menu_mark(3CURSES)

| NAME | menu_mark, set_menu_mark – menus mark string routines | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int set_menu_mark(MENU *menu, char *mark); char *menu_mark(MENU *menu);</pre> | | | | |
| DESCRIPTION | menus displays mark strings to distinguish selected items in a menu (or the current item in a single-valued menu). set_menu_mark() sets the mark string of menu to mark. menu_mark() returns a pointer to the mark string of menu. | | | | |
| RETURN VALUES | menu_mark() returns NULL on error. set_menu_mark() returns one of the following: E_OK The routine returned successfully. E_SYSTEM_ERROR System error. E_BAD_ARGUMENT An incorrect argument was passed to the routine. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: <table border="1" data-bbox="444 955 1414 1045"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), menus(3CURSES), attributes(5) | | | | |
| NOTES | The header <menu.h> automatically includes the headers <eti.h> and <curses.h>. | | | | |

| NAME | menu_opts, set_menu_opts, menu_opts_on, menu_opts_off – menus option routines | | | | | | | | | | | | |
|----------------------|--|----------------|--|----------------|---------------------------------------|------------|--------------------------------------|--------------|--|-------------|--|-------------|---|
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> OPTIONS menu_opts (MENU *menu) ; int set_menu_opts (MENU *menu, OPTIONS opts) ; int menu_opts_on (MENU *menu, OPTIONS opts) ; int menu_opts_off (MENU *menu, OPTIONS opts) ;</pre> | | | | | | | | | | | | |
| Menu Options | <p>set_menu_opts () turns on the named options for <i>menu</i> and turns off all other options. Options are boolean values that can be OR-ed together.</p> <p>menu_opts_on () turns on the named options for <i>menu</i>; no other option is changed.</p> <p>menu_opts_off () turns off the named options for <i>menu</i>; no other option is changed.</p> <p>menu_opts () returns the current options of <i>menu</i>.</p> <p>The following values can be OR'd together to create <i>opts</i>.</p> <table border="0"> <tr> <td>O_ONEVALUE</td> <td>Only one item can be selected from the menu.</td> </tr> <tr> <td>O_SHOWDESC</td> <td>Display the description of the items.</td> </tr> <tr> <td>O_ROWMAJOR</td> <td>Display the menu in row major order.</td> </tr> <tr> <td>O_IGNORECASE</td> <td>Ignore the case when pattern matching.</td> </tr> <tr> <td>O_SHOWMATCH</td> <td>Place the cursor within the item name when pattern matching.</td> </tr> <tr> <td>O_NONCYCLIC</td> <td>Make certain menu driver requests non-cyclic.</td> </tr> </table> | O_ONEVALUE | Only one item can be selected from the menu. | O_SHOWDESC | Display the description of the items. | O_ROWMAJOR | Display the menu in row major order. | O_IGNORECASE | Ignore the case when pattern matching. | O_SHOWMATCH | Place the cursor within the item name when pattern matching. | O_NONCYCLIC | Make certain menu driver requests non-cyclic. |
| O_ONEVALUE | Only one item can be selected from the menu. | | | | | | | | | | | | |
| O_SHOWDESC | Display the description of the items. | | | | | | | | | | | | |
| O_ROWMAJOR | Display the menu in row major order. | | | | | | | | | | | | |
| O_IGNORECASE | Ignore the case when pattern matching. | | | | | | | | | | | | |
| O_SHOWMATCH | Place the cursor within the item name when pattern matching. | | | | | | | | | | | | |
| O_NONCYCLIC | Make certain menu driver requests non-cyclic. | | | | | | | | | | | | |
| RETURN VALUES | <p>Except for menu_opts (), these routines return one of the following:</p> <table border="0"> <tr> <td>E_OK</td> <td>The routine returned successfully.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> <tr> <td>E_POSTED</td> <td>The menu is already posted.</td> </tr> </table> | E_OK | The routine returned successfully. | E_SYSTEM_ERROR | System error. | E_POSTED | The menu is already posted. | | | | | | |
| E_OK | The routine returned successfully. | | | | | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | | | | | |
| E_POSTED | The menu is already posted. | | | | | | | | | | | | |
| ATTRIBUTES | <p>See attributes(5) for descriptions of the following attributes:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | | | | | | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | | | | | | | |
| MT-Level | Unsafe | | | | | | | | | | | | |
| SEE ALSO | curses(3CURSES), menus(3CURSES), attributes(5) | | | | | | | | | | | | |
| NOTES | The header <menu.h> automatically includes the headers <eti.h> and <curses.h>. | | | | | | | | | | | | |

set_menu_pad(3CURSES)

| | | | | | | | |
|----------------------|--|------|------------------------------------|----------------|---------------|----------------|--|
| NAME | menu_attributes, set_menu_fore, menu_fore, set_menu_back, menu_back, set_menu_grey, menu_grey, set_menu_pad, menu_pad – control menus display attributes | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int set_menu_fore(MENU *menu, chtype attr); chtype menu_fore(MENU *menu); int set_menu_back(MENU *menu, chtype attr); chtype menu_back(MENU *menu); int set_menu_grey(MENU *menu, chtype attr); chtype menu_grey(MENU *menu); int set_menu_pad(MENU *menu, int pad); int menu_pad(MENU *menu);</pre> | | | | | | |
| DESCRIPTION | <p>set_menu_fore() sets the foreground attribute of <i>menu</i> — the display attribute for the current item (if selectable) on single-valued menus and for selected items on multi-valued menus. This display attribute is a curses library visual attribute. menu_fore() returns the foreground attribute of <i>menu</i>.</p> <p>set_menu_back() sets the background attribute of <i>menu</i> — the display attribute for unselected, yet selectable, items. This display attribute is a curses library visual attribute.</p> <p>set_menu_grey() sets the grey attribute of <i>menu</i> — the display attribute for nonselectable items in multi-valued menus. This display attribute is a curses library visual attribute. menu_grey() returns the grey attribute of <i>menu</i>.</p> <p>The pad character is the character that fills the space between the name and description of an item. set_menu_pad() sets the pad character for <i>menu</i> to <i>pad</i>. menu_pad() returns the pad character of <i>menu</i>.</p> | | | | | | |
| RETURN VALUES | <p>These routines return one of the following:</p> <table><tr><td>E_OK</td><td>The routine returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr><tr><td>E_BAD_ARGUMENT</td><td>An incorrect argument was passed to the routine.</td></tr></table> | E_OK | The routine returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An incorrect argument was passed to the routine. |
| E_OK | The routine returned successfully. | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

set_menu_pad(3CURSES)

SEE ALSO | curses(3CURSES), menus(3CURSES), attributes(5)

NOTES | The header `<menu.h>` automatically includes the headers `<eti.h>` and `<curses.h>`.

set_menu_pattern(3CURSES)

| NAME | menu_pattern, set_menu_pattern – set and get menus pattern match buffer | | | | | | | | |
|----------------------|--|----------------|------------------------------------|----------------|---------------|----------------|--|------------|--------------------------------|
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> char *menu_pattern (MENU *menu) ; int set_menu_pattern (MENU *menu, char *pat) ;</pre> | | | | | | | | |
| DESCRIPTION | Every menu has a pattern buffer to match entered data with menu items. set_menu_pattern() sets the pattern buffer to <i>pat</i> and tries to find the first item that matches the pattern. If it does, the matching item becomes the current item. If not, the current item does not change. menu_pattern() returns the string in the pattern buffer of <i>menu</i> . | | | | | | | | |
| RETURN VALUES | menu_pattern() returns NULL on error. set_menu_pattern() returns one of the following: <table><tr><td>E_OK</td><td>The routine returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr><tr><td>E_BAD_ARGUMENT</td><td>An incorrect argument was passed to the routine.</td></tr><tr><td>E_NO_MATCH</td><td>The character failed to match.</td></tr></table> | E_OK | The routine returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An incorrect argument was passed to the routine. | E_NO_MATCH | The character failed to match. |
| E_OK | The routine returned successfully. | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. | | | | | | | | |
| E_NO_MATCH | The character failed to match. | | | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | | | |
| MT-Level | Unsafe | | | | | | | | |
| SEE ALSO | curses(3CURSES), menus(3CURSES), attributes(5) | | | | | | | | |
| NOTES | The header <menu.h> automatically includes the headers <eti.h> and <curses.h>. | | | | | | | | |

| NAME | menu_win, set_menu_win, set_menu_sub, menu_sub, scale_menu – menus window and subwindow association routines | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int set_menu_win(MENU *menu, WINDOW *win); WINDOW *menu_win(MENU *menu); int set_menu_sub(MENU *menu, WINDOW *sub); WINDOW *menu_sub(MENU *menu); int scale_window(MENU *menu, int *rows, int *cols);</pre> | | | | |
| DESCRIPTION | <p>set_menu_win() sets the window of <i>menu</i> to <i>win</i>. menu_win() returns a pointer to the window of <i>menu</i>. set_menu_sub() sets the subwindow of <i>menu</i> to <i>sub</i>. menu_sub() returns a pointer to the subwindow of <i>menu</i>. scale_window() returns the minimum window size necessary for the subwindow of <i>menu</i>. <i>rows</i> and <i>cols</i> are pointers to the locations used to return the values.</p> | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <p>E_OK The routine returned successfully.</p> <p>E_SYSTEM_ERROR System error.</p> <p>E_BAD_ARGUMENT An incorrect argument was passed to the routine.</p> <p>E_POSTED The menu is already posted.</p> <p>E_NOT_CONNECTED No items are connected to the menu.</p> | | | | |
| ATTRIBUTES | <p>See attributes(5) for descriptions of the following attributes:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), menus(3CURSES), attributes(5) | | | | |
| NOTES | The header <menu.h> automatically includes the headers <eti.h> and <curses.h>. | | | | |

set_menu_term(3CURSES)

| | | | | | |
|----------------------|--|------|------------------------------------|----------------|---------------|
| NAME | menu_hook, set_item_init, item_init, set_item_term, item_term, set_menu_init, menu_init, set_menu_term, menu_term – assign application-specific routines for automatic invocation by menus | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int set_item_init(MENU *menu, void (*func)(MENU *)); int set_item_term(MENU *menu, void (*func)(MENU *)); void item_term(MENU *menu); int set_menu_init(MENU *menu, void (*func)(MENU *)); void menu_init(MENU *menu); int set_menu_term(MENU *menu, void (*func)(MENU *)); void menu_term(MENU *menu);</pre> | | | | |
| DESCRIPTION | <p>set_item_init() assigns the application-defined function to be called when the <i>menu</i> is posted and just after the current item changes. item_init() returns a pointer to the item initialization routine, if any, called when the <i>menu</i> is posted and just after the current item changes.</p> <p>set_item_term() assigns an application-defined function to be called when the <i>menu</i> is unposted and just before the current item changes. item_term() returns a pointer to the termination function, if any, called when the <i>menu</i> is unposted and just before the current item changes.</p> <p>set_menu_init() assigns an application-defined function to be called when the <i>menu</i> is posted and just after the top row changes on a posted menu. menu_init() returns a pointer to the menu initialization routine, if any, called when the <i>menu</i> is posted and just after the top row changes on a posted menu.</p> <p>set_menu_term() assigns an application-defined function to be called when the <i>menu</i> is unposted and just before the top row changes on a posted menu. menu_term() returns a pointer to the menu termination routine, if any, called when the <i>menu</i> is unposted and just before the top row changes on a posted menu.</p> | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <table><tr><td>E_OK</td><td>The routine returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr></table> | E_OK | The routine returned successfully. | E_SYSTEM_ERROR | System error. |
| E_OK | The routine returned successfully. | | | | |
| E_SYSTEM_ERROR | System error. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |

set_menu_term(3CURSES)

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), menus(3CURSES), attributes(5)

NOTES The header <menu.h> automatically includes the headers <eti.h> and <curses.h>.

set_menu_userptr(3CURSES)

NAME menu_userptr, set_menu_userptr – associate application data with menus

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

char *menu_userptr (MENU *menu) ;
int set_menu_userptr (MENU *menu, char *userptr) ;
```

DESCRIPTION Every menu has an associated user pointer that can be used to store relevant information. `set_menu_userptr()` sets the user pointer of `menu`. `menu_userptr()` returns the user pointer of `menu`.

RETURN VALUES `menu_userptr()` returns NULL on error.

`set_menu_userptr()` returns one of the following:

| | |
|----------------|------------------------------------|
| E_OK | The routine returned successfully. |
| E_SYSTEM_ERROR | System error. |

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3CURSES)`, `menus(3CURSES)`, `attributes(5)`

NOTES The header `<menu.h>` automatically includes the headers `<eti.h>` and `<curses.h>`.

| NAME | menu_win, set_menu_win, set_menu_sub, menu_sub, scale_menu – menus window and subwindow association routines | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int set_menu_win(MENU *menu, WINDOW *win); WINDOW *menu_win(MENU *menu); int set_menu_sub(MENU *menu, WINDOW *sub); WINDOW *menu_sub(MENU *menu); int scale_window(MENU *menu, int *rows, int *cols);</pre> | | | | |
| DESCRIPTION | <p>set_menu_win() sets the window of <i>menu</i> to <i>win</i>. menu_win() returns a pointer to the window of <i>menu</i>. set_menu_sub() sets the subwindow of <i>menu</i> to <i>sub</i>. menu_sub() returns a pointer to the subwindow of <i>menu</i>. scale_window() returns the minimum window size necessary for the subwindow of <i>menu</i>. <i>rows</i> and <i>cols</i> are pointers to the locations used to return the values.</p> | | | | |
| RETURN VALUES | <p>Routines that return pointers always return NULL on error. Routines that return an integer return one of the following:</p> <p>E_OK The routine returned successfully.</p> <p>E_SYSTEM_ERROR System error.</p> <p>E_BAD_ARGUMENT An incorrect argument was passed to the routine.</p> <p>E_POSTED The menu is already posted.</p> <p>E_NOT_CONNECTED No items are connected to the menu.</p> | | | | |
| ATTRIBUTES | <p>See attributes(5) for descriptions of the following attributes:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), menus(3CURSES), attributes(5) | | | | |
| NOTES | The header <menu.h> automatically includes the headers <eti.h> and <curses.h>. | | | | |

set_new_page(3CURSES)

NAME | form_new_page, set_new_page, new_page – forms pagination

SYNOPSIS |

```
cc [ flag... ] file... -lform -lcurses [ library... ]
#include <form.h>

int set_new_page(FIELD *field, int bool);
int new_page(FIELD *field);
```

DESCRIPTION | set_new_page() marks *field* as the beginning of a new page on the form.
new_page() returns a boolean value indicating whether or not *field* begins a new page of the form.

RETURN VALUES | new_page returns TRUE or FALSE.
set_new_page() returns one of the following:
E_OK The function returned successfully.
E_CONNECTED The field is already connected to a form.
E_SYSTEM_ERROR System error.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | curses(3CURSES), forms(3CURSES), attributes(5)

NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>.

set_panel_userptr(3CURSES)

NAME panel_userptr, set_panel_userptr – associate application data with a panels panel

SYNOPSIS

```
cc [ flag ... ] file ... -lpanel -lcurses [ library .. ]
#include <panel.h>

int set_panel_userptr(PANEL *panel, char *ptr);
char * panel_userptr(PANEL *panel);
```

DESCRIPTION Each panel has a user pointer available for maintaining relevant information.

set_panel_userptr() sets the user pointer of *panel* to *ptr*.

panel_userptr() returns the user pointer of *panel*.

RETURN VALUES set_panel_userptr returns OK if successful, ERR otherwise.

panel_userptr returns NULL if there is no user pointer assigned to *panel*.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), panels(3CURSES), attributes(5)

NOTES The header <panel.h> automatically includes the header <curses.h>.

setscrreg(3CURSES)

| | |
|--------------------|--|
| NAME | <code> curs_ouptops, clearok, idlok, idcok, immedok, leaveok, setscrreg, wsetscrreg, scrollok, nl, nonl</code> – curses terminal output option control routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int clearok(WINDOW *win, bool bf); int idlok(WINDOW *win, bool bf); void idcok(WINDOW *win, bool bf); void immedok(WINDOW *win, bool bf); int leaveok(WINDOW *win, bool bf); int setscrreg(int top, int bot); int wsetscrreg(WINDOW *win, int top, int bot); int scrollok(WINDOW *win, bool bf); int nl(void); int nonl(void);</pre> |
| DESCRIPTION | <p>These routines set options that deal with output within curses. All options are initially FALSE, unless otherwise stated. It is not necessary to turn these options off before calling <code>endwin()</code>.</p> <p>With the <code>clearok()</code> routine, if enabled (<i>bf</i> is TRUE), the next call to <code>wrefresh()</code> with this window will clear the screen completely and redraw the entire screen from scratch. This is useful when the contents of the screen are uncertain, or in some cases for a more pleasing visual effect. If the <i>win</i> argument to <code>clearok()</code> is the global variable <code>curscr()</code>, the next call to <code>wrefresh()</code> with any window causes the screen to be cleared and repainted from scratch.</p> <p>With the <code>idlok()</code> routine, if enabled (<i>bf</i> is TRUE), curses considers using the hardware insert/delete line feature of terminals so equipped. If disabled (<i>bf</i> is FALSE), curses very seldom uses this feature. (The insert/delete character feature is always considered.) This option should be enabled only if the application needs insert/delete line, for example, for a screen editor. It is disabled by default because insert/delete line tends to be visually annoying when used in applications where it isn't really needed. If insert/delete line cannot be used, curses redraws the changed portions of all lines.</p> <p>With the <code>idcok()</code> routine, if enabled (<i>bf</i> is TRUE), curses considers using the hardware insert/delete character feature of terminals so equipped. This is enabled by default.</p> <p>With the <code>immedok()</code> routine, if enabled (<i>bf</i> is TRUE), any change in the window image, such as the ones caused by <code>waddch()</code>, <code>wclrtoeol()</code>, <code>wscrl()</code>, etc., automatically cause a call to <code>wrefresh()</code>. However, it may degrade the performance considerably, due to repeated calls to <code>wrefresh()</code>. It is disabled by default.</p> |

setscrreg(3CURSES)

Normally, the hardware cursor is left at the location of the window cursor being refreshed. The `leaveok()` option allows the cursor to be left wherever the update happens to leave it. It is useful for applications where the cursor is not used, since it reduces the need for cursor motions. If possible, the cursor is made invisible when this option is enabled.

The `setscrreg()` and `wsetscrreg()` routines allow the application programmer to set a software scrolling region in a window. *top* and *bot* are the line numbers of the top and bottom margin of the scrolling region. (Line 0 is the top line of the window.) If this option and `scrollok()` are enabled, an attempt to move off the bottom margin line causes all lines in the scrolling region to scroll up one line. Only the text of the window is scrolled. (Note that this has nothing to do with the use of a physical scrolling region capability in the terminal, like that in the VT100. If `idlok()` is enabled and the terminal has either a scrolling region or insert/delete line capability, they will probably be used by the output routines.)

The `scrollok()` option controls what happens when the cursor of a window is moved off the edge of the window or scrolling region, either as a result of a newline action on the bottom line, or typing the last character of the last line. If disabled, (*bf* is FALSE), the cursor is left on the bottom line. If enabled, (*bf* is TRUE), `wrefresh()` is called on the window, and the physical terminal and window are scrolled up one line. (Note that in order to get the physical scrolling effect on the terminal, it is also necessary to call `idlok()`.)

The `nl()` and `nonl()` routines control whether newline is translated into carriage return and linefeed on output, and whether return is translated into newline on input. Initially, the translations do occur. By disabling these translations using `nonl()`, `curses` is able to make better use of the linefeed capability, resulting in faster cursor motion.

RETURN VALUES `setscrreg()` and `wsetscrreg()` return OK upon success and ERR upon failure. All other routines that return an integer always return OK.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_addch(3CURSES)`, `curs_clear(3CURSES)`, `curs_initscr(3CURSES)`, `curs_refresh(3CURSES)`, `curs_scroll(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `clearok()`, `leaveok()`, `scrollok()`, `idcok()`, `nl()`, `nonl()`, and `setscrreg()` may be macros.

setscrreg(3CURSES)

The `immedok()` routine is useful for windows that are used as terminal emulators.

| | |
|--------------------|---|
| NAME | clearok, idlok, leaveok, scrollok, setscrreg, wsetscrreg – terminal output control functions |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int clearok(WINDOW *win, bool bf); int idlok(WINDOW *win, bool bf); int leaveok(WINDOW *win, bool bf); int scrollok(WINDOW *win, bool bf); int setscrreg(int top, int bot); int wsetscrreg(WINDOW *win, int top, int bot);</pre> |
| DESCRIPTION | <p>These functions set options that deal with the output within Curses functions.</p> <p>The <code>clearok()</code> function assigns the value of <i>bf</i> to an internal flag in the specified window that governs clearing of the screen during a refresh. If, during a refresh operation on the specified window, the flag in <i>curscr</i> is TRUE or the flag in the specified window is TRUE, <code>clearok()</code> clears the screen, redraws it in its entirety, and sets the flag to FALSE in <i>curscr</i> and in the specified window. The initial state is unspecified</p> <p>The <code>idlok()</code> function specifies whether the implementation may use the hardware insert-line, delete-line, and scroll features of terminals so equipped. If <i>bf</i> is TRUE, use of these features is enabled. If <i>bf</i> is FALSE, use of these features is disabled and lines are instead redrawn as required. The initial state is FALSE.</p> <p>The <code>leaveok()</code> function controls the cursor position after a refresh operation. If <i>bf</i> is TRUE, refresh operations on the specified window may leave the terminal's cursor at an arbitrary position. If <i>bf</i> is FALSE, then at the end of any refresh operation, the terminal's cursor is positioned at the cursor position contained in the specified window. The initial state is FALSE.</p> <p>The <code>scrollok()</code> function controls the use of scrolling. If <i>bf</i> is TRUE, then scrolling is enabled for the specified window. If <i>bf</i> is FALSE, scrolling is disabled for the specified window. The initial state is FALSE.</p> <p>The <code>setscrreg()</code> and <code>wsetscrreg()</code> functions define a software scrolling region in the current or specified window. The <i>top</i> and <i>bottom</i> arguments are the line numbers of the first and last line defining the scrolling region. (Line 0 is the top line of the window.) If this option and <code>scrollok()</code> are enabled, an attempt to move off the last line of the margin causes all lines in the scrolling region to scroll one line in the</p> |

setscrreg(3XCURSES)

direction of the first line. Only characters in the window are scrolled. If a software scrolling region is set and `scrollok()` is not enabled, an attempt to move off the last line of the margin does not reposition any lines in the scrolling region.

PARAMETERS

win Is a pointer to a window.
bf Is a Boolean expression.
top Is the top line of the scrolling region (top of the window is line 0).
bot Is the bottom line of the scrolling region (top of the window is line 0).

RETURN VALUES

Upon successful completion, the `setscrreg()` and `wsetscrreg()` functions return OK. Otherwise, they return ERR.

The other functions always return OK.

ERRORS

No errors are defined.

USAGE

The only reason to enable the `idlok()` feature is to use scrolling to achieve the visual effect of motion of a partial window, such as for a screen editor. In other cases, the feature can be visually annoying.

The `leaveok()` option provides greater efficiency for applications that do not use the cursor.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO

`bkgdset(3XCURSES)`, `clear(3XCURSES)`, `doupdate(3XCURSES)`, `libcurses(3XCURSES)`, `scr1(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | curs_kernel, def_prog_mode, def_shell_mode, reset_prog_mode, reset_shell_mode, resetty, savetty, getsyx, setsyx, ripoffline, curs_set, napms – low-level curses routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int def_prog_mode(void); int def_shell_mode(void); int reset_prog_mode(void); int reset_shell_mode(void); int resetty(void); int savetty(void); int getsyx(int y, int x); int setsyx(int y, int x); int ripoffline(int line, int (*init)(WINDOW *, int)); int curs_set(int visibility); int napms(int ms);</pre> |
| DESCRIPTION | <p>The following routines give low-level access to various curses functionality. These routines typically are used inside library routines.</p> <p>The <code>def_prog_mode()</code> and <code>def_shell_mode()</code> routines save the current terminal modes as the “program” (in curses) or “shell” (not in curses) state for use by the <code>reset_prog_mode()</code> and <code>reset_shell_mode()</code> routines. This is done automatically by <code>initscr()</code>.</p> <p>The <code>reset_prog_mode()</code> and <code>reset_shell_mode()</code> routines restore the terminal to “program” (in curses) or “shell” (out of curses) state. These are done automatically by <code>endwin()</code> and, after an <code>endwin()</code>, by <code>doupdate()</code>, so they normally are not called.</p> <p>The <code>resetty()</code> and <code>savetty()</code> routines save and restore the state of the terminal modes. <code>savetty()</code> saves the current state in a buffer and <code>resetty()</code> restores the state to what it was at the last call to <code>savetty()</code>.</p> <p>With the <code>getsyx()</code> routine, the current coordinates of the virtual screen cursor are returned in <code>y</code> and <code>x</code>. If <code>leaveok()</code> is currently TRUE, then <code>-1,-1</code> is returned. If lines have been removed from the top of the screen, using <code>ripoffline()</code>, <code>y</code> and <code>x</code> include these lines; therefore, <code>y</code> and <code>x</code> should be used only as arguments for <code>setsyx()</code>.</p> <p>With the <code>setsyx()</code> routine, the virtual screen cursor is set to <code>y, x</code>. If <code>y</code> and <code>x</code> are both <code>-1</code>, then <code>leaveok()</code> is set. The two routines <code>getsyx()</code> and <code>setsyx()</code> are designed to be used by a library routine, which manipulates curses windows but does not</p> |

setsyx(3CURSES)

want to change the current position of the program's cursor. The library routine would call `getsyx()` at the beginning, do its manipulation of its own windows, do a `wnoutrefresh()` on its windows, call `setsyx()`, and then call `doupdate()`.

The `riporffline()` routine provides access to the same facility that `slk_init()` (see `curs_slk(3CURSES)`) uses to reduce the size of the screen. `riporffline()` must be called before `initscr()` or `newterm()` is called. If `line` is positive, a line is removed from the top of `stdscr()`; if `line` is negative, a line is removed from the bottom. When this is done inside `initscr()`, the routine `init()` (supplied by the user) is called with two arguments: a window pointer to the one-line window that has been allocated and an integer with the number of columns in the window. Inside this initialization routine, the integer variables `LINES` and `COLS` (defined in `<curses.h>`) are not guaranteed to be accurate and `wrefresh()` or `doupdate()` must not be called. It is allowable to call `wnoutrefresh()` during the initialization routine.

`riporffline()` can be called up to five times before calling `initscr()` or `newterm()`.

With the `curs_set()` routine, the cursor state is set to invisible, normal, or very visible for *visibility* equal to 0, 1, or 2 respectively. If the terminal supports the *visibility* requested, the previous *cursor* state is returned; otherwise, `ERR` is returned.

The `napms()` routine is used to sleep for *ms* milliseconds.

RETURN VALUES Except for `curs_set()`, these routines always return `OK`. `curs_set()` returns the previous cursor state, or `ERR` if the requested *visibility* is not supported.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_initscr(3CURSES)`, `curs_outopts(3CURSES)`, `curs_refresh(3CURSES)`, `curs_scr_dump(3CURSES)`, `curs_slk(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `getsyx()` is a macro, so an ampersand (&) is not necessary before the variables *y* and *x*.

| | |
|--------------------|--|
| NAME | curs_initscr, initscr, newterm, endwin, isendwin, set_term, delscreen – curses screen initialization and manipulation routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> WINDOW *initscr(void); int endwin(void); int isendwin(void); SCREEN *newterm(char *<i>type</i>, FILE *<i>outfd</i>, FILE *<i>infd</i>); SCREEN *set_term(SCREEN *<i>new</i>); void delscreen(SCREEN * <i>sp</i>);</pre> |
| DESCRIPTION | <p><code>initscr()</code> is almost always the first routine that should be called (the exceptions are <code>slk_init()</code>, <code>filter()</code>, <code>ripoffline()</code>, <code>use_env()</code> and, for multiple-terminal applications, <code>newterm()</code>.) This determines the terminal type and initializes all curses data structures. <code>initscr()</code> also causes the first call to <code>refresh()</code> to clear the screen. If errors occur, <code>initscr()</code> writes an appropriate error message to standard error and exits; otherwise, a pointer is returned to <code>stdscr()</code>. If the program needs an indication of error conditions, <code>newterm()</code> should be used instead of <code>initscr()</code>; <code>initscr()</code> should only be called once per application.</p> <p>A program that outputs to more than one terminal should use the <code>newterm()</code> routine for each terminal instead of <code>initscr()</code>. A program that needs an indication of error conditions, so it can continue to run in a line-oriented mode if the terminal cannot support a screen-oriented program, would also use this routine. The routine <code>newterm()</code> should be called once for each terminal. It returns a variable of type <code>SCREEN *</code> which should be saved as a reference to that terminal. The arguments are the <i>type</i> of the terminal to be used in place of <code>\$TERM</code>, a file pointer for output to the terminal, and another file pointer for input from the terminal (if <i>type</i> is <code>NULL</code>, <code>\$TERM</code> will be used). The program must also call <code>endwin()</code> for each terminal being used before exiting from curses. If <code>newterm()</code> is called more than once for the same terminal, the first terminal referred to must be the last one for which <code>endwin()</code> is called.</p> <p>A program should always call <code>endwin()</code> before exiting or escaping from curses mode temporarily. This routine restores tty modes, moves the cursor to the lower left-hand corner of the screen and resets the terminal into the proper non-visual mode. Calling <code>refresh()</code> or <code>doupdate()</code> after a temporary escape causes the program to resume visual mode.</p> <p>The <code>isendwin()</code> routine returns <code>TRUE</code> if <code>endwin()</code> has been called without any subsequent calls to <code>wrefresh()</code>, and <code>FALSE</code> otherwise.</p> |

set_term(3CURSES)

The `set_term()` routine is used to switch between different terminals. The screen reference `new` becomes the new current terminal. The previous terminal is returned by the routine. This is the only routine which manipulates `SCREEN` pointers; all other routines affect only the current terminal.

The `delscreen()` routine frees storage associated with the `SCREEN` data structure. The `endwin()` routine does not do this, so `delscreen()` should be called after `endwin()` if a particular `SCREEN` is no longer needed.

RETURN VALUES `endwin()` returns the integer `ERR` upon failure and `OK` upon successful completion.

Routines that return pointers always return `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_kernel(3CURSES)`, `curl_refresh(3CURSES)`, `curl_slk(3CURSES)`, `curl_util(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `initscr()` and `newterm()` may be macros.

| | |
|--------------------|--|
| NAME | curs_terminfo, setupterm, setterm, set_curterm, del_curterm, restartterm, tparm, tputs, putp, vidputs, vidattr, mvcur, tigetflag, tigetnum, tigetstr – curses interfaces to terminfo database |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> #include <term.h> int setupterm(char *term, int fildes, int *errret); int setterm(char *term); int set_curterm(TERMINAL *nterm); int del_curterm(TERMINAL *oterm); int restartterm(char *term, int fildes, int *errret); char *tparm(char *str, long int p1, long int p2, long int p3, long int p4, long int p5, long int p6, long int p7, long int p8, long int p9); int tputs(char *str, int affcnt, int (*putc)(char)); int putp(char *str); int vidputs(chtype attrs, int (*putc)(char)); int vidattr(chtype attrs); int mvcur(int oldrow, int oldcol, int newrow, int newcol); int tigetflag(char *capname); int tigetnum(char *capname); char *tigetstr(char *capname);</pre> |
| DESCRIPTION | <p>These low-level routines must be called by programs that have to deal directly with the <i>terminfo</i> database to handle certain terminal capabilities, such as programming function keys. For all other functionality, curses routines are more suitable and their use is recommended.</p> <p>Initially, <code>setupterm()</code> should be called. Note that <code>setupterm()</code> is automatically called by <code>initscr()</code> and <code>newterm()</code>. This defines the set of terminal-dependent variables (listed in <code>terminfo(4)</code>). The <i>terminfo</i> variables <code>lines</code> and <code>columns</code> are initialized by <code>setupterm()</code> as follows: If <code>use_env(FALSE)</code> has been called, values for <code>lines</code> and <code>columns</code> specified in <i>terminfo</i> are used. Otherwise, if the environment variables <code>LINES</code> and <code>COLUMNS</code> exist, their values are used. If these environment variables do not exist and the program is running in a window, the current window size is used. Otherwise, if the environment variables do not exist, the values for <code>lines</code> and <code>columns</code> specified in the <i>terminfo</i> database are used.</p> <p>The headers <code><curses.h></code> and <code><term.h></code> should be included (in this order) to get the definitions for these strings, numbers, and flags. Parameterized strings should be passed through <code>tparm()</code> to instantiate them. All <i>terminfo</i> strings (including the output</p> |

setterm(3CURSES)

of `tparm()` should be printed with `tputs()` or `putp()`. Call the `reset_shell_mode()` routine to restore the tty modes before exiting (see `curs_kernel(3CURSES)`). Programs which use cursor addressing should output `enter_ca_mode` upon startup and should output `exit_ca_mode` before exiting. Programs desiring shell escapes should call `reset_shell_mode` and output `exit_ca_mode` before the shell is called and should output `enter_ca_mode` and call `reset_prog_mode` after returning from the shell.

The `setupterm()` routine reads in the *terminfo* database, initializing the *terminfo* structures, but does not set up the output virtualization structures used by *curses*. The terminal type is the character string *term*; if *term* is null, the environment variable `TERM` is used. All output is to file descriptor *fildev* which is initialized for output. If *errret* is not null, then `setupterm()` returns `OK` or `ERR` and stores a status value in the integer pointed to by *errret*. A status of 1 in *errret* is normal, 0 means that the terminal could not be found, and -1 means that the *terminfo* database could not be found. If *errret* is null, `setupterm()` prints an error message upon finding an error and exits. Thus, the simplest call is:

```
setupterm((char *)0, 1, (int *)0);
```

which uses all the defaults and sends the output to `stdout`.

The `setterm()` routine is being replaced by `setupterm()`. The call:

```
setupterm(term, 1, (int *)0)
```

provides the same functionality as `setterm(term)`. The `setterm()` routine is included here for compatibility and is supported at Level 2.

The `set_curterm()` routine sets the variable `cur_term` to *nterm*, and makes all of the *terminfo* boolean, numeric, and string variables use the values from *nterm*.

The `del_curterm()` routine frees the space pointed to by *oterm* and makes it available for further use. If *oterm* is the same as `cur_term`, references to any of the *terminfo* boolean, numeric, and string variables thereafter may refer to invalid memory locations until another `setupterm()` has been called.

The `restartterm()` routine is similar to `setupterm()` and `initscr()`, except that it is called after restoring memory to a previous state. It assumes that the windows and the input and output options are the same as when memory was saved, but the terminal type and baud rate may be different.

The `tparm()` routine instantiates the string *str* with parameters *pi*. A pointer is returned to the result of *str* with the parameters applied.

The `tputs()` routine applies padding information to the string *str* and outputs it. The *str* must be a *terminfo* string variable or the return value from `tparm()`, `tgetstr()`, or `tgoto()`. *affcnt* is the number of lines affected, or 1 if not applicable. *putc* is a `putchar()`-like routine to which the characters are passed, one at a time.

setterm(3CURSES)

The `putp()` routine calls `tputs(str, 1, putchar)`. Note that the output of `putpA()` always goes to `stdout`, not to the *files* specified in `setupterm()`.

The `vidputs()` routine displays the string on the terminal in the video attribute mode *attrs*, which is any combination of the attributes listed in `curses(3CURSES)`. The characters are passed to the `putchar()`-like routine `putc()`.

The `vidattr()` routine is like the `vidputs()` routine, except that it outputs through `putchar()`.

The `mvcur()` routine provides low-level cursor motion.

The `tigetflag()`, `tigetnum()` and `tigetstr()` routines return the value of the capability corresponding to the *terminfo capname* passed to them, such as `xenl`.

With the `tigetflag()` routine, the value `-1` is returned if *capname* is not a boolean capability.

With the `tigetnum()` routine, the value `-2` is returned if *capname* is not a numeric capability.

With the `tigetstr()` routine, the value `(char *)-1` is returned if *capname* is not a string capability.

The *capname* for each capability is given in the table column entitled *capname* code in the capabilities section of `terminfo(4)`.

```
char *boolnames, *boolcodes, *boolfnames
char *numnames, *numcodes, *numfnames
char *strnames, *strcodes, *strfnames
```

These null-terminated arrays contain the *capnames*, the *termcap* codes, and the full C names, for each of the *terminfo* variables.

RETURN VALUES

All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

Routines that return pointers always return `NULL` on error.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_initscr(3CURSES)`, `curs_kernel(3CURSES)`, `curs_termcap(3CURSES)`, `curses(3CURSES)`, `putc(3C)`, `terminfo(4)`, `attributes(5)`

setterm(3CURSES)

NOTES | The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

The `setupterm()` routine should be used in place of `setterm()`.

Note that `vidattr()` and `vidputs()` may be macros.

| NAME | set_term – switch between terminals | | | | | | |
|----------------------|--|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> SCREEN *set_term(SCREEN *new);</pre> | | | | | | |
| PARAMETERS | <p><i>new</i> Is the new terminal to which the set_term() function will switch.</p> | | | | | | |
| DESCRIPTION | The set_term() function switches to the terminal specified by <i>new</i> and returns a screen reference to the previous terminal. Calls to subsequent X/Open Curses functions affect the new terminal. | | | | | | |
| RETURN VALUES | On success, the set_term() function returns a pointer to the previous screen. Otherwise, it returns a null pointer. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | libcurses(3XCURSES), attributes(5), standards(5) | | | | | | |

set_top_row(3CURSES)

NAME menu_item_current, set_current_item, current_item, set_top_row, top_row, item_index
– set and get current menus items

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

int set_current_item(MENU *menu, ITEM *item);
ITEM *current_item(MENU *menu);
int set_top_row(MENU *menu, int row);
int top_row(MENU *menu);
int item_index(ITEM *item);
```

DESCRIPTION The current item of a menu is the item where the cursor is currently positioned. `set_current_item()` sets the current item of *menu* to *item*. `current_item()` returns a pointer to the the current item in *menu*.

`set_top_row()` sets the top row of *menu* to *row*. The left-most item on the new top row becomes the current item. `top_row()` returns the number of the menu row currently displayed at the top of *menu*.

`item_index()` returns the index to the *item* in the item pointer array. The value of this index ranges from 0 through *N*-1, where *N* is the total number of items connected to the menu.

RETURN VALUES `current_item()` returns NULL on error.

`top_row()` and `index_item()` return -1 on error.

`set_current_item()` and `set_top_row()` return one of the following:

| | |
|-----------------|--|
| E_OK | The routine returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. |
| E_BAD_STATE | The routine was called from an initialization or termination function. |
| E_NOT_CONNECTED | No items are connected to the menu. |

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3CURSES)`, `menus(3CURSES)`, `attributes(5)`

`set_top_row(3CURSES)`

NOTES The header `<menu.h>` automatically includes the headers `<eti.h>` and `<curses.h>`.

setupterm(3CURSES)

| | |
|--------------------|--|
| NAME | <code> curs_terminfo, setupterm, setterm, set_curterm, del_curterm, restartterm, tparm, tputs, putp, vidputs, vidattr, mvcur, tigetflag, tigetnum, tigetstr</code> – curses interfaces to terminfo database |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> #include <term.h> int setupterm(char *term, int fildes, int *errret); int setterm(char *term); int set_curterm(TERMINAL *nterm); int del_curterm(TERMINAL *oterm); int restartterm(char *term, int fildes, int *errret); char *tparm(char *str, long int p1, long int p2, long int p3, long int p4, long int p5, long int p6, long int p7, long int p8, long int p9); int tputs(char *str, int affcnt, int (*putc)(char)); int putp(char *str); int vidputs(chtype attrs, int (*putc)(char)); int vidattr(chtype attrs); int mvcur(int oldrow, int oldcol, int newrow, int newcol); int tigetflag(char *capname); int tigetnum(char *capname); char *tigetstr(char *capname);</pre> |
| DESCRIPTION | <p>These low-level routines must be called by programs that have to deal directly with the <i>terminfo</i> database to handle certain terminal capabilities, such as programming function keys. For all other functionality, curses routines are more suitable and their use is recommended.</p> <p>Initially, <code>setupterm()</code> should be called. Note that <code>setupterm()</code> is automatically called by <code>initscr()</code> and <code>newterm()</code>. This defines the set of terminal-dependent variables (listed in <code>terminfo(4)</code>). The <i>terminfo</i> variables <code>lines</code> and <code>columns</code> are initialized by <code>setupterm()</code> as follows: If <code>use_env(FALSE)</code> has been called, values for <code>lines</code> and <code>columns</code> specified in <i>terminfo</i> are used. Otherwise, if the environment variables <code>LINES</code> and <code>COLUMNS</code> exist, their values are used. If these environment variables do not exist and the program is running in a window, the current window size is used. Otherwise, if the environment variables do not exist, the values for <code>lines</code> and <code>columns</code> specified in the <i>terminfo</i> database are used.</p> <p>The headers <code><curses.h></code> and <code><term.h></code> should be included (in this order) to get the definitions for these strings, numbers, and flags. Parameterized strings should be passed through <code>tparm()</code> to instantiate them. All <i>terminfo</i> strings (including the output</p> |

setupterm(3CURSES)

of `tparm()` should be printed with `tputs()` or `putp()`. Call the `reset_shell_mode()` routine to restore the tty modes before exiting (see `curs_kernel(3CURSES)`). Programs which use cursor addressing should output `enter_ca_mode` upon startup and should output `exit_ca_mode` before exiting. Programs desiring shell escapes should call `reset_shell_mode` and output `exit_ca_mode` before the shell is called and should output `enter_ca_mode` and call `reset_prog_mode` after returning from the shell.

The `setupterm()` routine reads in the *terminfo* database, initializing the *terminfo* structures, but does not set up the output virtualization structures used by *curses*. The terminal type is the character string *term*; if *term* is null, the environment variable `TERM` is used. All output is to file descriptor *fildev* which is initialized for output. If *errret* is not null, then `setupterm()` returns `OK` or `ERR` and stores a status value in the integer pointed to by *errret*. A status of 1 in *errret* is normal, 0 means that the terminal could not be found, and -1 means that the *terminfo* database could not be found. If *errret* is null, `setupterm()` prints an error message upon finding an error and exits. Thus, the simplest call is:

```
setupterm((char *)0, 1, (int *)0);
```

which uses all the defaults and sends the output to `stdout`.

The `setterm()` routine is being replaced by `setupterm()`. The call:

```
setupterm(term, 1, (int *)0)
```

provides the same functionality as `setterm(term)`. The `setterm()` routine is included here for compatibility and is supported at Level 2.

The `set_curterm()` routine sets the variable `cur_term` to *nterm*, and makes all of the *terminfo* boolean, numeric, and string variables use the values from *nterm*.

The `del_curterm()` routine frees the space pointed to by *oterm* and makes it available for further use. If *oterm* is the same as `cur_term`, references to any of the *terminfo* boolean, numeric, and string variables thereafter may refer to invalid memory locations until another `setupterm()` has been called.

The `restartterm()` routine is similar to `setupterm()` and `initscr()`, except that it is called after restoring memory to a previous state. It assumes that the windows and the input and output options are the same as when memory was saved, but the terminal type and baud rate may be different.

The `tparm()` routine instantiates the string *str* with parameters *pi*. A pointer is returned to the result of *str* with the parameters applied.

The `tputs()` routine applies padding information to the string *str* and outputs it. The *str* must be a *terminfo* string variable or the return value from `tparm()`, `tgetstr()`, or `tgoto()`. *affcnt* is the number of lines affected, or 1 if not applicable. *putc* is a `putchar()`-like routine to which the characters are passed, one at a time.

setupterm(3CURSES)

The `putp()` routine calls `tputs(str, 1, putchar)`. Note that the output of `putpA()` always goes to `stdout`, not to the *files* specified in `setupterm()`.

The `vidputs()` routine displays the string on the terminal in the video attribute mode *attrs*, which is any combination of the attributes listed in `curses(3CURSES)`. The characters are passed to the `putchar()`-like routine `putc()`.

The `vidattr()` routine is like the `vidputs()` routine, except that it outputs through `putchar()`.

The `mvcur()` routine provides low-level cursor motion.

The `tigetflag()`, `tigetnum()` and `tigetstr()` routines return the value of the capability corresponding to the *terminfo capname* passed to them, such as `xenl`.

With the `tigetflag()` routine, the value `-1` is returned if *capname* is not a boolean capability.

With the `tigetnum()` routine, the value `-2` is returned if *capname* is not a numeric capability.

With the `tigetstr()` routine, the value `(char *)-1` is returned if *capname* is not a string capability.

The *capname* for each capability is given in the table column entitled *capname code* in the capabilities section of `terminfo(4)`.

```
char *boolnames, *boolcodes, *boolfnames
char *numnames, *numcodes, *numfnames
char *strnames, *strcodes, *strfnames
```

These null-terminated arrays contain the *capnames*, the *termcap* codes, and the full C names, for each of the *terminfo* variables.

RETURN VALUES

All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

Routines that return pointers always return `NULL` on error.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_initscr(3CURSES)`, `curs_kernel(3CURSES)`, `curs_termcap(3CURSES)`, `curses(3CURSES)`, `putc(3C)`, `terminfo(4)`, `attributes(5)`

setupterm(3CURSES)

NOTES | The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

The `setupterm()` routine should be used in place of `setterm()`.

Note that `vidattr()` and `vidputs()` may be macros.

setupterm(3XCURSES)

| NAME | del_curterm, restartterm, set_curterm, setupterm – interfaces to the terminfo database | | | | | | | | |
|--------------------|--|-------|-------------|---|--------|---|-----------------------------|----|--------------------------------------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <term.h> int del_curterm(TERMINAL *oterm); int restartterm(char *term, int fildes, int *errret); TERMINAL *set_curterm(TERMINAL *nterm); int setupterm(char *term, int fildes, int *errret);</pre> | | | | | | | | |
| DESCRIPTION | <p>Within X/Open Curses, the <code>setupterm()</code> function is automatically called by the <code>initscr(3XC)</code> and <code>newterm(3XC)</code> functions. This function can be also be used outside of X/Open Curses when a program has to deal directly with the <code>terminfo</code> database to handle certain terminal capabilities. The use of appropriate X/Open Curses functions is recommended in all other situations.</p> <p>The <code>setupterm()</code> function loads terminal-dependent variables for the <code>terminfo</code> layer of X/Open Curses. The <code>setupterm()</code> function initializes the <code>terminfo</code> variables <code>lines</code> and <code>columns</code> such that if <code>use_env(FALSE)</code> has been called, the <code>terminfo</code> values assigned in the database are used regardless of the environmental variables <code>LINES</code> and <code>COLUMNS</code> or the program's window dimensions; when <code>use_env(TRUE)</code> has been called, which is the default, the environment variables <code>LINES</code> and <code>COLUMNS</code> are used, if they exist. If the environment variables do not exist and the program is running in a window, the current window size is used.</p> <p>The <code>term</code> parameter of <code>setupterm()</code> specifies the terminal; if null, terminal type is taken from the <code>TERM</code> environment variable. All output is sent to <code>fildes</code> which is initialized for output. If <code>errret</code> is not null, <code>OK</code> or <code>ERR</code> is returned and a status value is stored in the integer pointed to by <code>errret</code>. The following status values may be returned:</p> <table border="1"><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>1</td><td>Normal</td></tr><tr><td>0</td><td>Terminal could not be found</td></tr><tr><td>-1</td><td>terminfo database could not be found</td></tr></tbody></table> <p>If <code>errret</code> is null, an error message is printed, and the <code>setupterm()</code> function calls the <code>exit()</code> function with a non-zero parameter.</p> | Value | Description | 1 | Normal | 0 | Terminal could not be found | -1 | terminfo database could not be found |
| Value | Description | | | | | | | | |
| 1 | Normal | | | | | | | | |
| 0 | Terminal could not be found | | | | | | | | |
| -1 | terminfo database could not be found | | | | | | | | |

setupterm(3XCURSES)

The `set_curterm()` function sets the `cur_term` variable to `nterm`. The values from `nterm` as well as other state information for the terminal are used by X/Open Curses functions such as `beep(3XCURSES)`, `flash(3XCURSES)`, `mvcur(3XCURSES)`, `tigetflag(3XCURSES)`, `tigetstr(3XCURSES)`, and `tigetnum(3XCURSES)`.

The `del_curterm()` function frees the space pointed to by `oterm`. If `oterm` and the `cur_term` variable are the same, all Boolean, numeric, or string `terminfo` variables will refer to invalid memory locations until you call `setupterm()` and specify a new terminal type.

The `restartterm()` function assumes that a call to `setupterm()` has already been made (probably from `initscr()` or `newterm()`). It allows you to specify a new terminal type in `term` and updates the data returned by `baudrate(3XCURSES)` based on `fildev`. Other information created by the `initscr()`, `newterm()`, and `setupterm()` functions is preserved.

PARAMETERS

- oterm* Is the terminal type for which to free space.
- term* Is the terminal type for which variables are set.
- fildev* Is a file descriptor initialized for output.
- errret* Is a pointer to an integer in which the status value is stored.
- nterm* Is the new terminal to become the current terminal.

RETURN VALUES

On success, the `set_curterm()` function returns the previous value of `cur_term`. Otherwise, it returns a null pointer.

On success, the other functions return `OK`. Otherwise, they return `ERR`.

ERRORS

None.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO

`baudrate(3XCURSES)`, `beep(3XCURSES)`, `initscr(3XCURSES)`, `libcurses(3XCURSES)`, `mvcur(3XCURSES)`, `tigetflag(3XCURSES)`, `use_env(3XCURSES)`, `attributes(5)`, `standards(5)`

show_panel(3CURSES)

NAME panel_show, show_panel, hide_panel, panel_hidden – panels deck manipulation routines

SYNOPSIS

```
cc [ flag ... ] file ... -lpanel -lcurses [ library .. ]
#include <panel.h>

int show_panel (PANEL *panel) ;
int hide_panel (PANEL *panel) ;
int panel_hidden (PANEL *panel) ;
```

DESCRIPTION show_panel () makes *panel*, previously hidden, visible and places it on top of the deck of panels.

hide_panel () removes *panel* from the panel deck and, thus, hides it from view. The internal data structure of the panel is retained.

panel_hidden () returns TRUE (1) or FALSE (0) indicating whether or not *panel* is in the deck of panels.

RETURN VALUES show_panel () and hide_panel () return the integer OK upon successful completion or ERR upon error.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), panel_update(3CURSES), panels(3CURSES), attributes(5)

NOTES The header <panel.h> automatically includes the header <curses.h>.

| | |
|--------------------|--|
| NAME | curs_slk, slk_init, slk_set, slk_refresh, slk_noutrefresh, slk_label, slk_clear, slk_restore, slk_touch, slk_attron, slk_attrset, slk_attroff – curses soft label routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int slk_init(int <i>fmt</i>) ; int slk_set(int <i>labnum</i>, char *<i>label</i>, int <i>fmt</i>) ; int slk_refresh(void) ; int slk_noutrefresh(void) ; char *slk_label(int <i>labnum</i>) ; int slk_clear(void) ; int slk_restore(void) ; int slk_touch(void) ; int slk_attron(chtype <i>attrs</i>) ; int slk_attrset(chtype <i>attrs</i>) ; int slk_attroff(chtype <i>attrs</i>) ;</pre> |
| DESCRIPTION | <p>curses manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, <i>curses</i> takes over the bottom line of <code>stdscr</code>, reducing the size of <code>stdscr</code> and the variable <code>LINES</code>. <i>curses</i> standardizes on eight labels of up to eight characters each.</p> <p>To use soft labels, the <code>slk_init()</code> routine must be called before <code>initscr()</code> or <code>newterm()</code> is called. If <code>initscr()</code> eventually uses a line from <code>stdscr</code> to emulate the soft labels, then <code>fmt</code> determines how the labels are arranged on the screen. Setting <code>fmt</code> to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement.</p> <p>With the <code>slk_set()</code> routine, <i>labnum</i> is the label number, from 1 to 8. <i>label</i> is the string to be put on the label, up to eight characters in length. A null string or a null pointer sets up a blank label. <code>fmt</code> is either 0, 1, or 2, indicating whether the label is to be left-justified, centered, or right-justified, respectively, within the label.</p> <p>The <code>slk_refresh()</code> and <code>slk_noutrefresh()</code> routines correspond to the <code>wrefresh()</code> and <code>wnoutrefresh()</code> routines.</p> <p>With the <code>slk_label()</code> routine, the current label for label number <i>labnum</i> is returned with leading and trailing blanks stripped.</p> <p>With the <code>slk_clear()</code> routine, the soft labels are cleared from the screen.</p> <p>With the <code>slk_restore()</code> routine, the soft labels are restored to the screen after a <code>slk_clear()</code> is performed.</p> |

slk_attr(3CURSES)

With the `slk_touch()` routine, all the soft labels are forced to be output the next time a `slk_noutrefresh()` is performed.

The `slk_attron()`, `slk_attrset()`, and `slk_attr(3CURSES)` routines correspond to `attron()`, `attrset()`, and `attr(3CURSES)`. They have an effect only if soft labels are simulated on the bottom line of the screen.

RETURN VALUES Routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`slk_label()` returns `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `slk_attr(3CURSES)`, `slk_initscr(3CURSES)`, `slk_refresh(3CURSES)`, `slk_noutrefresh(3CURSES)`, `attributes(5)`

NOTES The header `< curses.h >` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.

| | |
|-------------------|--|
| NAME | slk_attroff, slk_attr_off, slk_attron, slk_attr_on, slk_attrset, slk_attr_set, slk_clear, slk_color, slk_init, slk_label, slk_noutrefresh, slk_refresh, slk_restore, slk_set, slk_touch, slk_wset – soft label functions |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int slk_attroff(const chtype attrs); int slk_attr_off(const attr_t attrs, void *opts); int slk_attron(const chtype attrs); int slk_attr_on(const attr_t attrs, void *opts); int slk_attrset(const chtype attrs); int slk_attr_set(const attr_t attrs, short color_pair_number, void *opts); int slk_clear(void); int slk_color(short color_pair_number); int slk_init(int fmt); char *slk_label(int labnum); int slk_noutrefresh(void); int slk_refresh(void); int slk_restore(void); int slk_set(int labnum, const char *label, int justify); int slk_touch(void); int slk_wset(int labnum, const wchar_t *label, int justify);</pre> |
| PARAMETERS | <p><i>attrs</i> are the window attributes to be added or removed.</p> <p><i>opts</i> Is reserved for future use. Currently, this must be a null pointer.</p> <p><i>color_pair_number</i> Is a color pair.</p> <p><i>fmt</i> Is the format of how the labels are arranged on the screen.</p> <p><i>labnum</i> Is the number of the soft label.</p> <p><i>label</i> Is the name to be given to a soft label.</p> <p><i>justify</i> Is a number indicating how to justify the label name.</p> |

slk_attr_off(3XCURSES)

DESCRIPTION

The Curses interface manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, Curses takes over the bottom line of *stdscr*, reducing the size of *stdscr* and the value of the `LINES` external variable. There can be up to eight labels of up to eight display columns each.

To use soft labels, `slk_init()` must be called before calling `initscr(3XCURSES)`, `newterm(3XCURSES)`, or `ripoffline(3XCURSES)`. If `initscr()` eventually uses a line from *stdscr* to emulate the soft labels, then *fmt* determines how the labels are arranged on the screen. Setting *fmt* to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement. Other values for *fmt* are unspecified.

The `slk_init()` function has the effect of calling `ripoffline()` to reserve one screen line to accommodate the requested format.

The `slk_set()` and `slk_wset()` functions specify the text of soft label number *labnum*, within the range from 1 to and including 8. The *label* argument is the string to be put the lable. With `slk_set()` and `slk_wset()`, the width of the label is limited to eight columns positions. A null string or a null pointer specifies a blank label. The *justify* argument can have the following values to indicate how to justify *label* within the space reserved for it:

- 0 Align the start of *label* with the start of the space
- 1 Center *label* within the space
- 2 Align the end of *label* with the end of the space

The `slk_refresh()` and `slk_noutrefresh()` functions correspond to the `wrefresh(3XCURSES)` and `wnoutrefresh(3XCURSES)` functions.

The `slk_label()` function obtains soft label number *labnum*.

The `slk_clear()` function immediately clears the soft labels from the screen.

The `slk_restore()` function immediately restores the soft labels to the screen after a call to `slk_clear()`.

The `slk_touch()` function forces all the soft labels to be output the next time `slk_refresh()` or `slk_noutrefresh()` is called.

The `slk_attron()`, `slk_attrset()`, and `slk_attroff()` functions correspond to the `attron(3XCURSES)`, `attrset(3XCURSES)`, and `attroff(3XCURSES)` functions. They have an effect only if soft labels are stimulated on the bottom line of the screen.

The `slk_attr_on()`, `slk_attr_off()`, `slk_attr_set()` and `slk_color()` functions correspond to the `attr_on(3XCURSES)`, `attr_off(3XCURSES)`, `attr_set(3XCURSES)`, and `color_set(3XCURSES)` functions. As a result, they support color and the attribute constants with the `WA_` prefix.

slk_attr_off(3XCURSES)

The *opts* argument is reserved for definition in a future release. Currently, the *opts* argument is a null pointer.

RETURN VALUES Upon successful completion, the `slk_label()` function returns the requested label with leading and trailing blanks stripped. Otherwise, it returns a null pointer.

Upon successful completion, the other functions return OK. Otherwise, they return ERR.

ERRORS No errors are defined.

USAGE When using multi-byte character sets, applications should check the width of the string by calling `mbstowcs(3C)` and then `wcswidth(3C)` before calling `slk_set()`. When using wide characters, applications should check the width of the string by calling `wcswidth()` before calling `slk_set()`.

Since the number of columns that a wide string will occupy is codeset-specific, call `wcwidth(3C)` and `wcswidth(3C)` to check the number of column positions in the string before calling `slk_wset()`.

Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attr_get(3XCURSES)`, `attroff(3XCURSES)`, `delscreen(3XCURSES)`, `libcurses(3XCURSES)`, `mbstowcs(3C)`, `ripcoffline(3XCURSES)`, `wcswidth(3C)`, `wcwidth(3C)`, `attributes(5)`, `standards(5)`

slk_attroff(3XCURSES)

| | | | | | | | | | | | | | | | |
|--------------------------|--|--------------|---|-------------|---|--------------------------|------------------|------------|---|---------------|----------------------------------|--------------|--|----------------|---|
| NAME | slk_attroff, slk_attr_off, slk_attron, slk_attr_on, slk_attrset, slk_attr_set, slk_clear, slk_color, slk_init, slk_label, slk_noutrefresh, slk_refresh, slk_restore, slk_set, slk_touch, slk_wset – soft label functions | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int slk_attroff(const chtype attrs); int slk_attr_off(const attr_t attrs, void *opts); int slk_attron(const chtype attrs); int slk_attr_on(const attr_t attrs, void *opts); int slk_attrset(const chtype attrs); int slk_attr_set(const attr_t attrs, short color_pair_number, void *opts); int slk_clear(void); int slk_color(short color_pair_number); int slk_init(int fmt); char *slk_label(int labnum); int slk_noutrefresh(void); int slk_refresh(void); int slk_restore(void); int slk_set(int labnum, const char *label, int justify); int slk_touch(void); int slk_wset(int labnum, const wchar_t *label, int justify);</pre> | | | | | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>attrs</i></td><td>are the window attributes to be added or removed.</td></tr><tr><td><i>opts</i></td><td>Is reserved for future use. Currently, this must be a null pointer.</td></tr><tr><td><i>color_pair_number</i></td><td>Is a color pair.</td></tr><tr><td><i>fmt</i></td><td>Is the format of how the labels are arranged on the screen.</td></tr><tr><td><i>labnum</i></td><td>Is the number of the soft label.</td></tr><tr><td><i>label</i></td><td>Is the name to be given to a soft label.</td></tr><tr><td><i>justify</i></td><td>Is a number indicating how to justify the label name.</td></tr></table> | <i>attrs</i> | are the window attributes to be added or removed. | <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | <i>color_pair_number</i> | Is a color pair. | <i>fmt</i> | Is the format of how the labels are arranged on the screen. | <i>labnum</i> | Is the number of the soft label. | <i>label</i> | Is the name to be given to a soft label. | <i>justify</i> | Is a number indicating how to justify the label name. |
| <i>attrs</i> | are the window attributes to be added or removed. | | | | | | | | | | | | | | |
| <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | | | | | | | | | | | | | | |
| <i>color_pair_number</i> | Is a color pair. | | | | | | | | | | | | | | |
| <i>fmt</i> | Is the format of how the labels are arranged on the screen. | | | | | | | | | | | | | | |
| <i>labnum</i> | Is the number of the soft label. | | | | | | | | | | | | | | |
| <i>label</i> | Is the name to be given to a soft label. | | | | | | | | | | | | | | |
| <i>justify</i> | Is a number indicating how to justify the label name. | | | | | | | | | | | | | | |

DESCRIPTION

The Curses interface manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, Curses takes over the bottom line of *stdscr*, reducing the size of *stdscr* and the value of the LINES external variable. There can be up to eight labels of up to eight display columns each.

To use soft labels, `slk_init()` must be called before calling `initscr(3XCURSES)`, `newterm(3XCURSES)`, or `ripoffline(3XCURSES)`. If `initscr()` eventually uses a line from *stdscr* to emulate the soft labels, then *fmt* determines how the labels are arranged on the screen. Setting *fmt* to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement. Other values for *fmt* are unspecified.

The `slk_init()` function has the effect of calling `ripoffline()` to reserve one screen line to accommodate the requested format.

The `slk_set()` and `slk_wset()` functions specify the text of soft label number *labnum*, within the range from 1 to and including 8. The *label* argument is the string to be put the lable. With `slk_set()` and `slk_wset()`, the width of the label is limited to eight columns positions. A null string or a null pointer specifies a blank label. The *justify* argument can have the following values to indicate how to justify *label* within the space reserved for it:

- 0 Align the start of *label* with the start of the space
- 1 Center *label* within the space
- 2 Align the end of *label* with the end of the space

The `slk_refresh()` and `slk_noutrefresh()` functions correspond to the `wrefresh(3XCURSES)` and `wnoutrefresh(3XCURSES)` functions.

The `slk_label()` function obtains soft label number *labnum*.

The `slk_clear()` function immediately clears the soft labels from the screen.

The `slk_restore()` function immediately restores the soft labels to the screen after a call to `slk_clear()`.

The `slk_touch()` function forces all the soft labels to be output the next time `slk_refresh()` or `slk_noutrefresh()` is called.

The `slk_attron()`, `slk_attrset()`, and `slk_attroff()` functions correspond to the `attron(3XCURSES)`, `attrset(3XCURSES)`, and `attroff(3XCURSES)` functions. They have an effect only if soft labels are stimulated on the bottom line of the screen.

The `slk_attr_on()`, `slk_attr_off()`, `slk_attr_set()` and `slk_color()` functions correspond to the `attr_on(3XCURSES)`, `attr_off(3XCURSES)`, `attr_set(3XCURSES)`, and `color_set(3XCURSES)` functions. As a result, they support color and the attribute constants with the `WA_` prefix.

slk_atroff(3XCURSES)

- The *opts* argument is reserved for definition in a future release. Currently, the *opts* argument is a null pointer.
- RETURN VALUES** Upon successful completion, the `slk_label()` function returns the requested label with leading and trailing blanks stripped. Otherwise, it returns a null pointer.
- Upon successful completion, the other functions return OK. Otherwise, they return ERR.
- ERRORS** No errors are defined.
- USAGE** When using multi-byte character sets, applications should check the width of the string by calling `mbstowcs(3C)` and then `wcswidth(3C)` before calling `slk_set()`. When using wide characters, applications should check the width of the string by calling `wcswidth()` before calling `slk_set()`.
- Since the number of columns that a wide string will occupy is codeset-specific, call `wcwidth(3C)` and `wcswidth(3C)` to check the number of column positions in the string before calling `slk_wset()`.
- Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.
- ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:
- | ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |
- SEE ALSO** `attr_get(3XCURSES)`, `attroff(3XCURSES)`, `delscreen(3XCURSES)`, `libcurses(3XCURSES)`, `mbstowcs(3C)`, `ripoffline(3XCURSES)`, `wcswidth(3C)`, `wcwidth(3C)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | curs_slk, slk_init, slk_set, slk_refresh, slk_noutrefresh, slk_label, slk_clear, slk_restore, slk_touch, slk_attron, slk_attrset, slk_attroff – curses soft label routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int slk_init(int <i>fmt</i>) ; int slk_set(int <i>labnum</i>, char *<i>label</i>, int <i>fmt</i>) ; int slk_refresh(void) ; int slk_noutrefresh(void) ; char *slk_label(int <i>labnum</i>) ; int slk_clear(void) ; int slk_restore(void) ; int slk_touch(void) ; int slk_attron(chtype <i>attrs</i>) ; int slk_attrset(chtype <i>attrs</i>) ; int slk_attroff(chtype <i>attrs</i>) ;</pre> |
| DESCRIPTION | <p>curses manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, <i>curses</i> takes over the bottom line of <code>stdscr</code>, reducing the size of <code>stdscr</code> and the variable <code>LINES</code>. <i>curses</i> standardizes on eight labels of up to eight characters each.</p> <p>To use soft labels, the <code>slk_init()</code> routine must be called before <code>initscr()</code> or <code>newterm()</code> is called. If <code>initscr()</code> eventually uses a line from <code>stdscr</code> to emulate the soft labels, then <code>fmt</code> determines how the labels are arranged on the screen. Setting <code>fmt</code> to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement.</p> <p>With the <code>slk_set()</code> routine, <i>labnum</i> is the label number, from 1 to 8. <i>label</i> is the string to be put on the label, up to eight characters in length. A null string or a null pointer sets up a blank label. <code>fmt</code> is either 0, 1, or 2, indicating whether the label is to be left-justified, centered, or right-justified, respectively, within the label.</p> <p>The <code>slk_refresh()</code> and <code>slk_noutrefresh()</code> routines correspond to the <code>wrefresh()</code> and <code>wnoutrefresh()</code> routines.</p> <p>With the <code>slk_label()</code> routine, the current label for label number <i>labnum</i> is returned with leading and trailing blanks stripped.</p> <p>With the <code>slk_clear()</code> routine, the soft labels are cleared from the screen.</p> <p>With the <code>slk_restore()</code> routine, the soft labels are restored to the screen after a <code>slk_clear()</code> is performed.</p> |

slk_attron(3CURSES)

With the `slk_touch()` routine, all the soft labels are forced to be output the next time a `slk_noutrefresh()` is performed.

The `slk_attron()`, `slk_attrset()`, and `slk_attroff()` routines correspond to `attron()`, `attrset()`, and `attroff()`. They have an effect only if soft labels are simulated on the bottom line of the screen.

RETURN VALUES Routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`slk_label()` returns `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_attr(3CURSES)`, `curl_initscr(3CURSES)`, `curl_refresh(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.

| | |
|-------------------|--|
| NAME | slk_attroff, slk_attr_off, slk_attron, slk_attr_on, slk_attrset, slk_attr_set, slk_clear, slk_color, slk_init, slk_label, slk_noutrefresh, slk_refresh, slk_restore, slk_set, slk_touch, slk_wset – soft label functions |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int slk_attroff(const chtype attrs); int slk_attr_off(const attr_t attrs, void *opts); int slk_attron(const chtype attrs); int slk_attr_on(const attr_t attrs, void *opts); int slk_attrset(const chtype attrs); int slk_attr_set(const attr_t attrs, short color_pair_number, void *opts); int slk_clear(void); int slk_color(short color_pair_number); int slk_init(int fmt); char *slk_label(int labnum); int slk_noutrefresh(void); int slk_refresh(void); int slk_restore(void); int slk_set(int labnum, const char *label, int justify); int slk_touch(void); int slk_wset(int labnum, const wchar_t *label, int justify);</pre> |
| PARAMETERS | <p><i>attrs</i> are the window attributes to be added or removed.</p> <p><i>opts</i> Is reserved for future use. Currently, this must be a null pointer.</p> <p><i>color_pair_number</i> Is a color pair.</p> <p><i>fmt</i> Is the format of how the labels are arranged on the screen.</p> <p><i>labnum</i> Is the number of the soft label.</p> <p><i>label</i> Is the name to be given to a soft label.</p> <p><i>justify</i> Is a number indicating how to justify the label name.</p> |

slk_attr_on(3XCURSES)

DESCRIPTION

The Curses interface manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, Curses takes over the bottom line of *stdscr*, reducing the size of *stdscr* and the value of the LINES external variable. There can be up to eight labels of up to eight display columns each.

To use soft labels, `slk_init()` must be called before calling `initscr(3XCURSES)`, `newterm(3XCURSES)`, or `ripoffline(3XCURSES)`. If `initscr()` eventually uses a line from *stdscr* to emulate the soft labels, then *fmt* determines how the labels are arranged on the screen. Setting *fmt* to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement. Other values for *fmt* are unspecified.

The `slk_init()` function has the effect of calling `ripoffline()` to reserve one screen line to accommodate the requested format.

The `slk_set()` and `slk_wset()` functions specify the text of soft label number *labnum*, within the range from 1 to and including 8. The *label* argument is the string to be put the lable. With `slk_set()` and `slk_wset()`, the width of the label is limited to eight columns positions. A null string or a null pointer specifies a blank label. The *justify* argument can have the following values to indicate how to justify *label* within the space reserved for it:

- 0 Align the start of *label* with the start of the space
- 1 Center *label* within the space
- 2 Align the end of *label* with the end of the space

The `slk_refresh()` and `slk_noutrefresh()` functions correspond to the `wrefresh(3XCURSES)` and `wnoutrefresh(3XCURSES)` functions.

The `slk_label()` function obtains soft label number *labnum*.

The `slk_clear()` function immediately clears the soft labels from the screen.

The `slk_restore()` function immediately restores the soft labels to the screen after a call to `slk_clear()`.

The `slk_touch()` function forces all the soft labels to be output the next time `slk_refresh()` or `slk_noutrefresh()` is called.

The `slk_attron()`, `slk_attrset()`, and `slk_attroff()` functions correspond to the `attron(3XCURSES)`, `attrset(3XCURSES)`, and `attroff(3XCURSES)` functions. They have an effect only if soft labels are stimulated on the bottom line of the screen.

The `slk_attr_on()`, `slk_attr_off()`, `slk_attr_set()` and `slk_color()` functions correspond to the `attr_on(3XCURSES)`, `attr_off(3XCURSES)`, `attr_set(3XCURSES)`, and `color_set(3XCURSES)` functions. As a result, they support color and the attribute constants with the `WA_` prefix.

slk_attr_on(3XCURSES)

The *opts* argument is reserved for definition in a future release. Currently, the *opts* argument is a null pointer.

RETURN VALUES Upon successful completion, the `slk_label()` function returns the requested label with leading and trailing blanks stripped. Otherwise, it returns a null pointer.

Upon successful completion, the other functions return OK. Otherwise, they return ERR.

ERRORS No errors are defined.

USAGE When using multi-byte character sets, applications should check the width of the string by calling `mbstowcs(3C)` and then `wcswidth(3C)` before calling `slk_set()`. When using wide characters, applications should check the width of the string by calling `wcswidth()` before calling `slk_set()`.

Since the number of columns that a wide string will occupy is codeset-specific, call `wcwidth(3C)` and `wcswidth(3C)` to check the number of column positions in the string before calling `slk_wset()`.

Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attr_get(3XCURSES)`, `attroff(3XCURSES)`, `delscreen(3XCURSES)`, `libcurses(3XCURSES)`, `mbstowcs(3C)`, `ripoffline(3XCURSES)`, `wcswidth(3C)`, `wcwidth(3C)`, `attributes(5)`, `standards(5)`

slk_attron(3XCURSES)

| | | | | | | | | | | | | | | | |
|--------------------------|--|--------------|---|-------------|---|--------------------------|------------------|------------|---|---------------|----------------------------------|--------------|--|----------------|---|
| NAME | slk_attroff, slk_attr_off, slk_attron, slk_attr_on, slk_attrset, slk_attr_set, slk_clear, slk_color, slk_init, slk_label, slk_noutrefresh, slk_refresh, slk_restore, slk_set, slk_touch, slk_wset – soft label functions | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int slk_attroff(const chtype attrs); int slk_attr_off(const attr_t attrs, void *opts); int slk_attron(const chtype attrs); int slk_attr_on(const attr_t attrs, void *opts); int slk_attrset(const chtype attrs); int slk_attr_set(const attr_t attrs, short color_pair_number, void *opts); int slk_clear(void); int slk_color(short color_pair_number); int slk_init(int fmt); char *slk_label(int labnum); int slk_noutrefresh(void); int slk_refresh(void); int slk_restore(void); int slk_set(int labnum, const char *label, int justify); int slk_touch(void); int slk_wset(int labnum, const wchar_t *label, int justify);</pre> | | | | | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>attrs</i></td><td>are the window attributes to be added or removed.</td></tr><tr><td><i>opts</i></td><td>Is reserved for future use. Currently, this must be a null pointer.</td></tr><tr><td><i>color_pair_number</i></td><td>Is a color pair.</td></tr><tr><td><i>fmt</i></td><td>Is the format of how the labels are arranged on the screen.</td></tr><tr><td><i>labnum</i></td><td>Is the number of the soft label.</td></tr><tr><td><i>label</i></td><td>Is the name to be given to a soft label.</td></tr><tr><td><i>justify</i></td><td>Is a number indicating how to justify the label name.</td></tr></table> | <i>attrs</i> | are the window attributes to be added or removed. | <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | <i>color_pair_number</i> | Is a color pair. | <i>fmt</i> | Is the format of how the labels are arranged on the screen. | <i>labnum</i> | Is the number of the soft label. | <i>label</i> | Is the name to be given to a soft label. | <i>justify</i> | Is a number indicating how to justify the label name. |
| <i>attrs</i> | are the window attributes to be added or removed. | | | | | | | | | | | | | | |
| <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | | | | | | | | | | | | | | |
| <i>color_pair_number</i> | Is a color pair. | | | | | | | | | | | | | | |
| <i>fmt</i> | Is the format of how the labels are arranged on the screen. | | | | | | | | | | | | | | |
| <i>labnum</i> | Is the number of the soft label. | | | | | | | | | | | | | | |
| <i>label</i> | Is the name to be given to a soft label. | | | | | | | | | | | | | | |
| <i>justify</i> | Is a number indicating how to justify the label name. | | | | | | | | | | | | | | |

DESCRIPTION | The Curses interface manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, Curses takes over the bottom line of *stdscr*, reducing the size of *stdscr* and the value of the `LINES` external variable. There can be up to eight labels of up to eight display columns each.

To use soft labels, `slk_init()` must be called before calling `initscr(3XCURSES)`, `newterm(3XCURSES)`, or `ripoffline(3XCURSES)`. If `initscr()` eventually uses a line from *stdscr* to emulate the soft labels, then *fmt* determines how the labels are arranged on the screen. Setting *fmt* to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement. Other values for *fmt* are unspecified.

The `slk_init()` function has the effect of calling `ripoffline()` to reserve one screen line to accommodate the requested format.

The `slk_set()` and `slk_wset()` functions specify the text of soft label number *labnum*, within the range from 1 to and including 8. The *label* argument is the string to be put the lable. With `slk_set()` and `slk_wset()`, the width of the label is limited to eight columns positions. A null string or a null pointer specifies a blank label. The *justify* argument can have the following values to indicate how to justify *label* within the space reserved for it:

- 0 Align the start of *label* with the start of the space
- 1 Center *label* within the space
- 2 Align the end of *label* with the end of the space

The `slk_refresh()` and `slk_noutrefresh()` functions correspond to the `wrefresh(3XCURSES)` and `wnoutrefresh(3XCURSES)` functions.

The `slk_label()` function obtains soft label number *labnum*.

The `slk_clear()` function immediately clears the soft labels from the screen.

The `slk_restore()` function immediately restores the soft labels to the screen after a call to `slk_clear()`.

The `slk_touch()` function forces all the soft labels to be output the next time `slk_refresh()` or `slk_noutrefresh()` is called.

The `slk_attron()`, `slk_attrset()`, and `slk_attroff()` functions correspond to the `attron(3XCURSES)`, `attrset(3XCURSES)`, and `attroff(3XCURSES)` functions. They have an effect only if soft labels are stimulated on the bottom line of the screen.

The `slk_attr_on()`, `slk_attr_off()`, `slk_attr_set()` and `slk_color()` functions correspond to the `attr_on(3XCURSES)`, `attr_off(3XCURSES)`, `attr_set(3XCURSES)`, and `color_set(3XCURSES)` functions. As a result, they support color and the attribute constants with the `WA_` prefix.

slk_atron(3XCURSES)

The *opts* argument is reserved for definition in a future release. Currently, the *opts* argument is a null pointer.

RETURN VALUES Upon successful completion, the `slk_label()` function returns the requested label with leading and trailing blanks stripped. Otherwise, it returns a null pointer.

Upon successful completion, the other functions return OK. Otherwise, they return ERR.

ERRORS No errors are defined.

USAGE When using multi-byte character sets, applications should check the width of the string by calling `mbstowcs(3C)` and then `wcswidth(3C)` before calling `slk_set()`. When using wide characters, applications should check the width of the string by calling `wcswidth()` before calling `slk_set()`.

Since the number of columns that a wide string will occupy is codeset-specific, call `wcwidth(3C)` and `wcswidth(3C)` to check the number of column positions in the string before calling `slk_wset()`.

Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attr_get(3XCURSES)`, `attroff(3XCURSES)`, `delscreen(3XCURSES)`, `libcurses(3XCURSES)`, `mbstowcs(3C)`, `ripoffline(3XCURSES)`, `wcswidth(3C)`, `wcwidth(3C)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | curs_slk, slk_init, slk_set, slk_refresh, slk_noutrefresh, slk_label, slk_clear, slk_restore, slk_touch, slk_atron, slk_attrset, slk_attroff – curses soft label routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int slk_init(int <i>fmt</i>); int slk_set(int <i>labnum</i>, char *<i>label</i>, int <i>fmt</i>); int slk_refresh(void); int slk_noutrefresh(void); char *slk_label(int <i>labnum</i>); int slk_clear(void); int slk_restore(void); int slk_touch(void); int slk_atron(chtype <i>attrs</i>); int slk_attrset(chtype <i>attrs</i>); int slk_attroff(chtype <i>attrs</i>);</pre> |
| DESCRIPTION | <p>curses manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, <i>curses</i> takes over the bottom line of <code>stdscr</code>, reducing the size of <code>stdscr</code> and the variable <code>LINES</code>. <i>curses</i> standardizes on eight labels of up to eight characters each.</p> <p>To use soft labels, the <code>slk_init()</code> routine must be called before <code>initscr()</code> or <code>newterm()</code> is called. If <code>initscr()</code> eventually uses a line from <code>stdscr</code> to emulate the soft labels, then <code>fmt</code> determines how the labels are arranged on the screen. Setting <code>fmt</code> to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement.</p> <p>With the <code>slk_set()</code> routine, <i>labnum</i> is the label number, from 1 to 8. <i>label</i> is the string to be put on the label, up to eight characters in length. A null string or a null pointer sets up a blank label. <code>fmt</code> is either 0, 1, or 2, indicating whether the label is to be left-justified, centered, or right-justified, respectively, within the label.</p> <p>The <code>slk_refresh()</code> and <code>slk_noutrefresh()</code> routines correspond to the <code>wrefresh()</code> and <code>wnoutrefresh()</code> routines.</p> <p>With the <code>slk_label()</code> routine, the current label for label number <i>labnum</i> is returned with leading and trailing blanks stripped.</p> <p>With the <code>slk_clear()</code> routine, the soft labels are cleared from the screen.</p> <p>With the <code>slk_restore()</code> routine, the soft labels are restored to the screen after a <code>slk_clear()</code> is performed.</p> |

slk_attrset(3CURSES)

With the `slk_touch()` routine, all the soft labels are forced to be output the next time a `slk_noutrefresh()` is performed.

The `slk_attron()`, `slk_attrset()`, and `slk_attroff()` routines correspond to `attron()`, `attrset()`, and `attroff()`. They have an effect only if soft labels are simulated on the bottom line of the screen.

RETURN VALUES Routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`slk_label()` returns `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_attr(3CURSES)`, `curl_initscr(3CURSES)`, `curl_refresh(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.

| | | | | | | | | | | | | | | | |
|--------------------------|--|--------------|---|-------------|---|--------------------------|------------------|------------|---|---------------|----------------------------------|--------------|--|----------------|---|
| NAME | slk_attroff, slk_attr_off, slk_attron, slk_attr_on, slk_attrset, slk_attr_set, slk_clear, slk_color, slk_init, slk_label, slk_noutrefresh, slk_refresh, slk_restore, slk_set, slk_touch, slk_wset – soft label functions | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int slk_attroff(const chtype attrs); int slk_attr_off(const attr_t attrs, void *opts); int slk_attron(const chtype attrs); int slk_attr_on(const attr_t attrs, void *opts); int slk_attrset(const chtype attrs); int slk_attr_set(const attr_t attrs, short color_pair_number, void *opts); int slk_clear(void); int slk_color(short color_pair_number); int slk_init(int fmt); char *slk_label(int labnum); int slk_noutrefresh(void); int slk_refresh(void); int slk_restore(void); int slk_set(int labnum, const char *label, int justify); int slk_touch(void); int slk_wset(int labnum, const wchar_t *label, int justify);</pre> | | | | | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>attrs</i></td> <td>are the window attributes to be added or removed.</td> </tr> <tr> <td><i>opts</i></td> <td>Is reserved for future use. Currently, this must be a null pointer.</td> </tr> <tr> <td><i>color_pair_number</i></td> <td>Is a color pair.</td> </tr> <tr> <td><i>fmt</i></td> <td>Is the format of how the labels are arranged on the screen.</td> </tr> <tr> <td><i>labnum</i></td> <td>Is the number of the soft label.</td> </tr> <tr> <td><i>label</i></td> <td>Is the name to be given to a soft label.</td> </tr> <tr> <td><i>justify</i></td> <td>Is a number indicating how to justify the label name.</td> </tr> </table> | <i>attrs</i> | are the window attributes to be added or removed. | <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | <i>color_pair_number</i> | Is a color pair. | <i>fmt</i> | Is the format of how the labels are arranged on the screen. | <i>labnum</i> | Is the number of the soft label. | <i>label</i> | Is the name to be given to a soft label. | <i>justify</i> | Is a number indicating how to justify the label name. |
| <i>attrs</i> | are the window attributes to be added or removed. | | | | | | | | | | | | | | |
| <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | | | | | | | | | | | | | | |
| <i>color_pair_number</i> | Is a color pair. | | | | | | | | | | | | | | |
| <i>fmt</i> | Is the format of how the labels are arranged on the screen. | | | | | | | | | | | | | | |
| <i>labnum</i> | Is the number of the soft label. | | | | | | | | | | | | | | |
| <i>label</i> | Is the name to be given to a soft label. | | | | | | | | | | | | | | |
| <i>justify</i> | Is a number indicating how to justify the label name. | | | | | | | | | | | | | | |

slk_attr_set(3XCURSES)

DESCRIPTION

The Curses interface manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, Curses takes over the bottom line of *stdscr*, reducing the size of *stdscr* and the value of the `LINES` external variable. There can be up to eight labels of up to eight display columns each.

To use soft labels, `slk_init()` must be called before calling `initscr(3XCURSES)`, `newterm(3XCURSES)`, or `ripoffline(3XCURSES)`. If `initscr()` eventually uses a line from *stdscr* to emulate the soft labels, then *fmt* determines how the labels are arranged on the screen. Setting *fmt* to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement. Other values for *fmt* are unspecified.

The `slk_init()` function has the effect of calling `ripoffline()` to reserve one screen line to accommodate the requested format.

The `slk_set()` and `slk_wset()` functions specify the text of soft label number *labnum*, within the range from 1 to and including 8. The *label* argument is the string to be put the lable. With `slk_set()` and `slk_wset()`, the width of the label is limited to eight columns positions. A null string or a null pointer specifies a blank label. The *justify* argument can have the following values to indicate how to justify *label* within the space reserved for it:

- 0 Align the start of *label* with the start of the space
- 1 Center *label* within the space
- 2 Align the end of *label* with the end of the space

The `slk_refresh()` and `slk_noutrefresh()` functions correspond to the `wrefresh(3XCURSES)` and `wnoutrefresh(3XCURSES)` functions.

The `slk_label()` function obtains soft label number *labnum*.

The `slk_clear()` function immediately clears the soft labels from the screen.

The `slk_restore()` function immediately restores the soft labels to the screen after a call to `slk_clear()`.

The `slk_touch()` function forces all the soft labels to be output the next time `slk_refresh()` or `slk_noutrefresh()` is called.

The `slk_attron()`, `slk_attrset()`, and `slk_attrtuff()` functions correspond to the `attron(3XCURSES)`, `attrset(3XCURSES)`, and `attroff(3XCURSES)` functions. They have an effect only if soft labels are stimulated on the bottom line of the screen.

The `slk_attr_on()`, `slk_attr_off()`, `slk_attr_set()` and `slk_color()` functions correspond to the `attr_on(3XCURSES)`, `attr_off(3XCURSES)`, `attr_set(3XCURSES)`, and `color_set(3XCURSES)` functions. As a result, they support color and the attribute constants with the `WA_` prefix.

slk_attr_set(3XCURSES)

The *opts* argument is reserved for definition in a future release. Currently, the *opts* argument is a null pointer.

RETURN VALUES Upon successful completion, the `slk_label()` function returns the requested label with leading and trailing blanks stripped. Otherwise, it returns a null pointer.

Upon successful completion, the other functions return OK. Otherwise, they return ERR.

ERRORS No errors are defined.

USAGE When using multi-byte character sets, applications should check the width of the string by calling `mbstowcs(3C)` and then `wcswidth(3C)` before calling `slk_set()`. When using wide characters, applications should check the width of the string by calling `wcswidth()` before calling `slk_set()`.

Since the number of columns that a wide string will occupy is codeset-specific, call `wcwidth(3C)` and `wcswidth(3C)` to check the number of column positions in the string before calling `slk_wset()`.

Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attr_get(3XCURSES)`, `attroff(3XCURSES)`, `delscreen(3XCURSES)`, `libcurses(3XCURSES)`, `mbstowcs(3C)`, `ripcoffline(3XCURSES)`, `wcswidth(3C)`, `wcwidth(3C)`, `attributes(5)`, `standards(5)`

slk_attrset(3XCURSES)

| | | | | | | | | | | | | | | | |
|--------------------------|--|--------------|---|-------------|---|--------------------------|------------------|------------|---|---------------|----------------------------------|--------------|--|----------------|---|
| NAME | slk_attrset, slk_attr_off, slk_attron, slk_attr_on, slk_attrset, slk_attr_set, slk_clear, slk_color, slk_init, slk_label, slk_noutrefresh, slk_refresh, slk_restore, slk_set, slk_touch, slk_wset – soft label functions | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int slk_attrset(const chtype attrs); int slk_attr_off(const attr_t attrs, void *opts); int slk_attron(const chtype attrs); int slk_attr_on(const attr_t attrs, void *opts); int slk_attrset(const chtype attrs); int slk_attr_set(const attr_t attrs, short color_pair_number, void *opts); int slk_clear(void); int slk_color(short color_pair_number); int slk_init(int fmt); char *slk_label(int labnum); int slk_noutrefresh(void); int slk_refresh(void); int slk_restore(void); int slk_set(int labnum, const char *label, int justify); int slk_touch(void); int slk_wset(int labnum, const wchar_t *label, int justify);</pre> | | | | | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>attrs</i></td><td>are the window attributes to be added or removed.</td></tr><tr><td><i>opts</i></td><td>Is reserved for future use. Currently, this must be a null pointer.</td></tr><tr><td><i>color_pair_number</i></td><td>Is a color pair.</td></tr><tr><td><i>fmt</i></td><td>Is the format of how the labels are arranged on the screen.</td></tr><tr><td><i>labnum</i></td><td>Is the number of the soft label.</td></tr><tr><td><i>label</i></td><td>Is the name to be given to a soft label.</td></tr><tr><td><i>justify</i></td><td>Is a number indicating how to justify the label name.</td></tr></table> | <i>attrs</i> | are the window attributes to be added or removed. | <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | <i>color_pair_number</i> | Is a color pair. | <i>fmt</i> | Is the format of how the labels are arranged on the screen. | <i>labnum</i> | Is the number of the soft label. | <i>label</i> | Is the name to be given to a soft label. | <i>justify</i> | Is a number indicating how to justify the label name. |
| <i>attrs</i> | are the window attributes to be added or removed. | | | | | | | | | | | | | | |
| <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | | | | | | | | | | | | | | |
| <i>color_pair_number</i> | Is a color pair. | | | | | | | | | | | | | | |
| <i>fmt</i> | Is the format of how the labels are arranged on the screen. | | | | | | | | | | | | | | |
| <i>labnum</i> | Is the number of the soft label. | | | | | | | | | | | | | | |
| <i>label</i> | Is the name to be given to a soft label. | | | | | | | | | | | | | | |
| <i>justify</i> | Is a number indicating how to justify the label name. | | | | | | | | | | | | | | |

DESCRIPTION

The Curses interface manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, Curses takes over the bottom line of *stdscr*, reducing the size of *stdscr* and the value of the `LINES` external variable. There can be up to eight labels of up to eight display columns each.

To use soft labels, `slk_init()` must be called before calling `initscr(3XCURSES)`, `newterm(3XCURSES)`, or `ripoffline(3XCURSES)`. If `initscr()` eventually uses a line from *stdscr* to emulate the soft labels, then *fmt* determines how the labels are arranged on the screen. Setting *fmt* to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement. Other values for *fmt* are unspecified.

The `slk_init()` function has the effect of calling `ripoffline()` to reserve one screen line to accommodate the requested format.

The `slk_set()` and `slk_wset()` functions specify the text of soft label number *labnum*, within the range from 1 to and including 8. The *label* argument is the string to be put the lable. With `slk_set()` and `slk_wset()`, the width of the label is limited to eight columns positions. A null string or a null pointer specifies a blank label. The *justify* argument can have the following values to indicate how to justify *label* within the space reserved for it:

- 0 Align the start of *label* with the start of the space
- 1 Center *label* within the space
- 2 Align the end of *label* with the end of the space

The `slk_refresh()` and `slk_noutrefresh()` functions correspond to the `wrefresh(3XCURSES)` and `wnoutrefresh(3XCURSES)` functions.

The `slk_label()` function obtains soft label number *labnum*.

The `slk_clear()` function immediately clears the soft labels from the screen.

The `slk_restore()` function immediately restores the soft labels to the screen after a call to `slk_clear()`.

The `slk_touch()` function forces all the soft labels to be output the next time `slk_refresh()` or `slk_noutrefresh()` is called.

The `slk_attron()`, `slk_attrset()`, and `slk_attroff()` functions correspond to the `attron(3XCURSES)`, `attrset(3XCURSES)`, and `attroff(3XCURSES)` functions. They have an effect only if soft labels are stimulated on the bottom line of the screen.

The `slk_attr_on()`, `slk_attr_off()`, `slk_attr_set()` and `slk_color()` functions correspond to the `attr_on(3XCURSES)`, `attr_off(3XCURSES)`, `attr_set(3XCURSES)`, and `color_set(3XCURSES)` functions. As a result, they support color and the attribute constants with the `WA_` prefix.

slk_attrset(3XCURSES)

- The *opts* argument is reserved for definition in a future release. Currently, the *opts* argument is a null pointer.
- RETURN VALUES** Upon successful completion, the `slk_label()` function returns the requested label with leading and trailing blanks stripped. Otherwise, it returns a null pointer.
- Upon successful completion, the other functions return OK. Otherwise, they return ERR.
- ERRORS** No errors are defined.
- USAGE** When using multi-byte character sets, applications should check the width of the string by calling `mbstowcs(3C)` and then `wcswidth(3C)` before calling `slk_set()`. When using wide characters, applications should check the width of the string by calling `wcswidth()` before calling `slk_set()`.
- Since the number of columns that a wide string will occupy is codeset-specific, call `wcwidth(3C)` and `wcswidth(3C)` to check the number of column positions in the string before calling `slk_wset()`.
- Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.
- ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:
- | ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |
- SEE ALSO** `attr_get(3XCURSES)`, `attroff(3XCURSES)`, `delscreen(3XCURSES)`, `libcurses(3XCURSES)`, `mbstowcs(3C)`, `ripoffline(3XCURSES)`, `wcswidth(3C)`, `wcwidth(3C)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | curs_slk, slk_init, slk_set, slk_refresh, slk_noutrefresh, slk_label, slk_clear, slk_restore, slk_touch, slk_attron, slk_attrset, slk_attroff – curses soft label routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int slk_init(int fmt); int slk_set(int labnum, char *label, int fmt); int slk_refresh(void); int slk_noutrefresh(void); char *slk_label(int labnum); int slk_clear(void); int slk_restore(void); int slk_touch(void); int slk_attron(chtype attrs); int slk_attrset(chtype attrs); int slk_attroff(chtype attrs);</pre> |
| DESCRIPTION | <p>curses manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, <i>curses</i> takes over the bottom line of <code>stdscr</code>, reducing the size of <code>stdscr</code> and the variable <code>LINES</code>. <i>curses</i> standardizes on eight labels of up to eight characters each.</p> <p>To use soft labels, the <code>slk_init()</code> routine must be called before <code>initscr()</code> or <code>newterm()</code> is called. If <code>initscr()</code> eventually uses a line from <code>stdscr</code> to emulate the soft labels, then <code>fmt</code> determines how the labels are arranged on the screen. Setting <code>fmt</code> to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement.</p> <p>With the <code>slk_set()</code> routine, <i>labnum</i> is the label number, from 1 to 8. <i>label</i> is the string to be put on the label, up to eight characters in length. A null string or a null pointer sets up a blank label. <code>fmt</code> is either 0, 1, or 2, indicating whether the label is to be left-justified, centered, or right-justified, respectively, within the label.</p> <p>The <code>slk_refresh()</code> and <code>slk_noutrefresh()</code> routines correspond to the <code>wrefresh()</code> and <code>wnoutrefresh()</code> routines.</p> <p>With the <code>slk_label()</code> routine, the current label for label number <i>labnum</i> is returned with leading and trailing blanks stripped.</p> <p>With the <code>slk_clear()</code> routine, the soft labels are cleared from the screen.</p> <p>With the <code>slk_restore()</code> routine, the soft labels are restored to the screen after a <code>slk_clear()</code> is performed.</p> |

slk_clear(3CURSES)

With the `slk_touch()` routine, all the soft labels are forced to be output the next time a `slk_noutrefresh()` is performed.

The `slk_attron()`, `slk_attrset()`, and `slk_attroff()` routines correspond to `attron()`, `attrset()`, and `attroff()`. They have an effect only if soft labels are simulated on the bottom line of the screen.

RETURN VALUES Routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`slk_label()` returns `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_attr(3CURSES)`, `curl_initscr(3CURSES)`, `curl_refresh(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.

| | |
|-------------------|---|
| NAME | slk_attroff, slk_attr_off, slk_attron, slk_attr_on, slk_attrset, slk_attr_set, slk_clear, slk_color, slk_init, slk_label, slk_noutrefresh, slk_refresh, slk_restore, slk_set, slk_touch, slk_wset – soft label functions |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int slk_attroff(const chtype <i>attrs</i>); int slk_attr_off(const attr_t <i>attrs</i>, void *<i>opts</i>); int slk_attron(const chtype <i>attrs</i>); int slk_attr_on(const attr_t <i>attrs</i>, void *<i>opts</i>); int slk_attrset(const chtype <i>attrs</i>); int slk_attr_set(const attr_t <i>attrs</i>, short <i>color_pair_number</i>, void *<i>opts</i>); int slk_clear(void); int slk_color(short <i>color_pair_number</i>); int slk_init(int <i>fmt</i>); char *slk_label(int <i>labnum</i>); int slk_noutrefresh(void); int slk_refresh(void); int slk_restore(void); int slk_set(int <i>labnum</i>, const char *<i>label</i>, int <i>justify</i>); int slk_touch(void); int slk_wset(int <i>labnum</i>, const wchar_t *<i>label</i>, int <i>justify</i>);</pre> |
| PARAMETERS | <p><i>attrs</i> are the window attributes to be added or removed.</p> <p><i>opts</i> Is reserved for future use. Currently, this must be a null pointer.</p> <p><i>color_pair_number</i> Is a color pair.</p> <p><i>fmt</i> Is the format of how the labels are arranged on the screen.</p> <p><i>labnum</i> Is the number of the soft label.</p> <p><i>label</i> Is the name to be given to a soft label.</p> <p><i>justify</i> Is a number indicating how to justify the label name.</p> |

slk_clear(3XCURSES)

DESCRIPTION

The Curses interface manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, Curses takes over the bottom line of *stdscr*, reducing the size of *stdscr* and the value of the LINES external variable. There can be up to eight labels of up to eight display columns each.

To use soft labels, *slk_init()* must be called before calling *initscr(3XCURSES)*, *newterm(3XCURSES)*, or *ripoffline(3XCURSES)*. If *initscr()* eventually uses a line from *stdscr* to emulate the soft labels, then *fmt* determines how the labels are arranged on the screen. Setting *fmt* to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement. Other values for *fmt* are unspecified.

The *slk_init()* function has the effect of calling *ripoffline()* to reserve one screen line to accommodate the requested format.

The *slk_set()* and *slk_wset()* functions specify the text of soft label number *labnum*, within the range from 1 to and including 8. The *label* argument is the string to be put the lable. With *slk_set()* and *slk_wset()*, the width of the label is limited to eight columns positions. A null string or a null pointer specifies a blank label. The *justify* argument can have the following values to indicate how to justify *label* within the space reserved for it:

- 0 Align the start of *label* with the start of the space
- 1 Center *label* within the space
- 2 Align the end of *label* with the end of the space

The *slk_refresh()* and *slk_noutrefresh()* functions correspond to the *wrefresh(3XCURSES)* and *wnoutrefresh(3XCURSES)* functions.

The *slk_label()* function obtains soft label number *labnum*.

The *slk_clear()* function immediately clears the soft labels from the screen.

The *slk_restore()* function immediately restores the soft labels to the screen after a call to *slk_clear()*.

The *slk_touch()* function forces all the soft labels to be output the next time *slk_refresh()* or *slk_noutrefresh()* is called.

The *slk_attron()*, *slk_attrset()*, and *slk_attroff()* functions correspond to the *attron(3XCURSES)*, *attrset(3XCURSES)*, and *attroff(3XCURSES)* functions. They have an effect only if soft labels are stimulated on the bottom line of the screen.

The *slk_attr_on()*, *slk_attr_off()*, *slk_attr_set()* and *slk_color()* functions correspond to the *attr_on(3XCURSES)*, *attr_off(3XCURSES)*, *attr_set(3XCURSES)*, and *color_set(3XCURSES)* functions. As a result, they support color and the attribute constants with the WA_ prefix.

slk_clear(3XCURSES)

The *opts* argument is reserved for definition in a future release. Currently, the *opts* argument is a null pointer.

RETURN VALUES Upon successful completion, the `slk_label()` function returns the requested label with leading and trailing blanks stripped. Otherwise, it returns a null pointer.

Upon successful completion, the other functions return OK. Otherwise, they return ERR.

ERRORS No errors are defined.

USAGE When using multi-byte character sets, applications should check the width of the string by calling `mbstowcs(3C)` and then `wcswidth(3C)` before calling `slk_set()`. When using wide characters, applications should check the width of the string by calling `wcswidth()` before calling `slk_set()`.

Since the number of columns that a wide string will occupy is codeset-specific, call `wcwidth(3C)` and `wcswidth(3C)` to check the number of column positions in the string before calling `slk_wset()`.

Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attr_get(3XCURSES)`, `attroff(3XCURSES)`, `delscreen(3XCURSES)`, `libcurses(3XCURSES)`, `mbstowcs(3C)`, `ripcoffline(3XCURSES)`, `wcswidth(3C)`, `wcwidth(3C)`, `attributes(5)`, `standards(5)`

slk_color(3XCURSES)

| | | | | | | | | | | | | | | | |
|--------------------------|--|--------------|---|-------------|---|--------------------------|------------------|------------|---|---------------|----------------------------------|--------------|--|----------------|---|
| NAME | slk_attroff, slk_attr_off, slk_attron, slk_attr_on, slk_attrset, slk_attr_set, slk_clear, slk_color, slk_init, slk_label, slk_noutrefresh, slk_refresh, slk_restore, slk_set, slk_touch, slk_wset – soft label functions | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int slk_attroff(const chtype attrs); int slk_attr_off(const attr_t attrs, void *opts); int slk_attron(const chtype attrs); int slk_attr_on(const attr_t attrs, void *opts); int slk_attrset(const chtype attrs); int slk_attr_set(const attr_t attrs, short color_pair_number, void *opts); int slk_clear(void); int slk_color(short color_pair_number); int slk_init(int fmt); char *slk_label(int labnum); int slk_noutrefresh(void); int slk_refresh(void); int slk_restore(void); int slk_set(int labnum, const char *label, int justify); int slk_touch(void); int slk_wset(int labnum, const wchar_t *label, int justify);</pre> | | | | | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>attrs</i></td><td>are the window attributes to be added or removed.</td></tr><tr><td><i>opts</i></td><td>Is reserved for future use. Currently, this must be a null pointer.</td></tr><tr><td><i>color_pair_number</i></td><td>Is a color pair.</td></tr><tr><td><i>fmt</i></td><td>Is the format of how the labels are arranged on the screen.</td></tr><tr><td><i>labnum</i></td><td>Is the number of the soft label.</td></tr><tr><td><i>label</i></td><td>Is the name to be given to a soft label.</td></tr><tr><td><i>justify</i></td><td>Is a number indicating how to justify the label name.</td></tr></table> | <i>attrs</i> | are the window attributes to be added or removed. | <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | <i>color_pair_number</i> | Is a color pair. | <i>fmt</i> | Is the format of how the labels are arranged on the screen. | <i>labnum</i> | Is the number of the soft label. | <i>label</i> | Is the name to be given to a soft label. | <i>justify</i> | Is a number indicating how to justify the label name. |
| <i>attrs</i> | are the window attributes to be added or removed. | | | | | | | | | | | | | | |
| <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | | | | | | | | | | | | | | |
| <i>color_pair_number</i> | Is a color pair. | | | | | | | | | | | | | | |
| <i>fmt</i> | Is the format of how the labels are arranged on the screen. | | | | | | | | | | | | | | |
| <i>labnum</i> | Is the number of the soft label. | | | | | | | | | | | | | | |
| <i>label</i> | Is the name to be given to a soft label. | | | | | | | | | | | | | | |
| <i>justify</i> | Is a number indicating how to justify the label name. | | | | | | | | | | | | | | |

DESCRIPTION | The Curses interface manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, Curses takes over the bottom line of *stdscr*, reducing the size of *stdscr* and the value of the `LINES` external variable. There can be up to eight labels of up to eight display columns each.

To use soft labels, `slk_init()` must be called before calling `initscr(3XCURSES)`, `newterm(3XCURSES)`, or `ripoffline(3XCURSES)`. If `initscr()` eventually uses a line from *stdscr* to emulate the soft labels, then *fmt* determines how the labels are arranged on the screen. Setting *fmt* to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement. Other values for *fmt* are unspecified.

The `slk_init()` function has the effect of calling `ripoffline()` to reserve one screen line to accommodate the requested format.

The `slk_set()` and `slk_wset()` functions specify the text of soft label number *labnum*, within the range from 1 to and including 8. The *label* argument is the string to be put the lable. With `slk_set()` and `slk_wset()`, the width of the label is limited to eight columns positions. A null string or a null pointer specifies a blank label. The *justify* argument can have the following values to indicate how to justify *label* within the space reserved for it:

- 0 Align the start of *label* with the start of the space
- 1 Center *label* within the space
- 2 Align the end of *label* with the end of the space

The `slk_refresh()` and `slk_noutrefresh()` functions correspond to the `wrefresh(3XCURSES)` and `wnoutrefresh(3XCURSES)` functions.

The `slk_label()` function obtains soft label number *labnum*.

The `slk_clear()` function immediately clears the soft labels from the screen.

The `slk_restore()` function immediately restores the soft labels to the screen after a call to `slk_clear()`.

The `slk_touch()` function forces all the soft labels to be output the next time `slk_refresh()` or `slk_noutrefresh()` is called.

The `slk_attron()`, `slk_attrset()`, and `slk_attroff()` functions correspond to the `attron(3XCURSES)`, `attrset(3XCURSES)`, and `attroff(3XCURSES)` functions. They have an effect only if soft labels are stimulated on the bottom line of the screen.

The `slk_attr_on()`, `slk_attr_off()`, `slk_attr_set()` and `slk_color()` functions correspond to the `attr_on(3XCURSES)`, `attr_off(3XCURSES)`, `attr_set(3XCURSES)`, and `color_set(3XCURSES)` functions. As a result, they support color and the attribute constants with the `WA_` prefix.

slk_color(3XCURSES)

The *opts* argument is reserved for definition in a future release. Currently, the *opts* argument is a null pointer.

RETURN VALUES Upon successful completion, the `slk_label()` function returns the requested label with leading and trailing blanks stripped. Otherwise, it returns a null pointer.

Upon successful completion, the other functions return OK. Otherwise, they return ERR.

ERRORS No errors are defined.

USAGE When using multi-byte character sets, applications should check the width of the string by calling `mbstowcs(3C)` and then `wcswidth(3C)` before calling `slk_set()`. When using wide characters, applications should check the width of the string by calling `wcswidth()` before calling `slk_set()`.

Since the number of columns that a wide string will occupy is codeset-specific, call `wcwidth(3C)` and `wcswidth(3C)` to check the number of column positions in the string before calling `slk_wset()`.

Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attr_get(3XCURSES)`, `attroff(3XCURSES)`, `delscreen(3XCURSES)`, `libcurses(3XCURSES)`, `mbstowcs(3C)`, `ripoffline(3XCURSES)`, `wcswidth(3C)`, `wcwidth(3C)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | curs_slk, slk_init, slk_set, slk_refresh, slk_noutrefresh, slk_label, slk_clear, slk_restore, slk_touch, slk_attron, slk_attrset, slk_attroff – curses soft label routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int slk_init(int <i>fmt</i>) ; int slk_set(int <i>labnum</i>, char *<i>label</i>, int <i>fmt</i>) ; int slk_refresh(void) ; int slk_noutrefresh(void) ; char *slk_label(int <i>labnum</i>) ; int slk_clear(void) ; int slk_restore(void) ; int slk_touch(void) ; int slk_attron(chtype <i>attrs</i>) ; int slk_attrset(chtype <i>attrs</i>) ; int slk_attroff(chtype <i>attrs</i>) ;</pre> |
| DESCRIPTION | <p>curses manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, <i>curses</i> takes over the bottom line of <code>stdscr</code>, reducing the size of <code>stdscr</code> and the variable <code>LINES</code>. <i>curses</i> standardizes on eight labels of up to eight characters each.</p> <p>To use soft labels, the <code>slk_init()</code> routine must be called before <code>initscr()</code> or <code>newterm()</code> is called. If <code>initscr()</code> eventually uses a line from <code>stdscr</code> to emulate the soft labels, then <i>fmt</i> determines how the labels are arranged on the screen. Setting <i>fmt</i> to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement.</p> <p>With the <code>slk_set()</code> routine, <i>labnum</i> is the label number, from 1 to 8. <i>label</i> is the string to be put on the label, up to eight characters in length. A null string or a null pointer sets up a blank label. <i>fmt</i> is either 0, 1, or 2, indicating whether the label is to be left-justified, centered, or right-justified, respectively, within the label.</p> <p>The <code>slk_refresh()</code> and <code>slk_noutrefresh()</code> routines correspond to the <code>wrefresh()</code> and <code>wnoutrefresh()</code> routines.</p> <p>With the <code>slk_label()</code> routine, the current label for label number <i>labnum</i> is returned with leading and trailing blanks stripped.</p> <p>With the <code>slk_clear()</code> routine, the soft labels are cleared from the screen.</p> <p>With the <code>slk_restore()</code> routine, the soft labels are restored to the screen after a <code>slk_clear()</code> is performed.</p> |

slk_init(3CURSES)

With the `slk_touch()` routine, all the soft labels are forced to be output the next time a `slk_noutrefresh()` is performed.

The `slk_attron()`, `slk_attrset()`, and `slk_attroff()` routines correspond to `attron()`, `attrset()`, and `attroff()`. They have an effect only if soft labels are simulated on the bottom line of the screen.

RETURN VALUES Routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`slk_label()` returns `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_attr(3CURSES)`, `curl_initscr(3CURSES)`, `curl_refresh(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.

| | |
|-------------------|---|
| NAME | slk_attroff, slk_attr_off, slk_attron, slk_attr_on, slk_attrset, slk_attr_set, slk_clear, slk_color, slk_init, slk_label, slk_noutrefresh, slk_refresh, slk_restore, slk_set, slk_touch, slk_wset – soft label functions |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int slk_attroff(const chtype <i>attrs</i>); int slk_attr_off(const attr_t <i>attrs</i>, void *<i>opts</i>); int slk_attron(const chtype <i>attrs</i>); int slk_attr_on(const attr_t <i>attrs</i>, void *<i>opts</i>); int slk_attrset(const chtype <i>attrs</i>); int slk_attr_set(const attr_t <i>attrs</i>, short <i>color_pair_number</i>, void *<i>opts</i>); int slk_clear(void); int slk_color(short <i>color_pair_number</i>); int slk_init(int <i>fmt</i>); char *slk_label(int <i>labnum</i>); int slk_noutrefresh(void); int slk_refresh(void); int slk_restore(void); int slk_set(int <i>labnum</i>, const char *<i>label</i>, int <i>justify</i>); int slk_touch(void); int slk_wset(int <i>labnum</i>, const wchar_t *<i>label</i>, int <i>justify</i>);</pre> |
| PARAMETERS | <p><i>attrs</i> are the window attributes to be added or removed.</p> <p><i>opts</i> Is reserved for future use. Currently, this must be a null pointer.</p> <p><i>color_pair_number</i> Is a color pair.</p> <p><i>fmt</i> Is the format of how the labels are arranged on the screen.</p> <p><i>labnum</i> Is the number of the soft label.</p> <p><i>label</i> Is the name to be given to a soft label.</p> <p><i>justify</i> Is a number indicating how to justify the label name.</p> |

slk_init(3XCURSES)

DESCRIPTION

The Curses interface manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, Curses takes over the bottom line of *stdscr*, reducing the size of *stdscr* and the value of the `LINES` external variable. There can be up to eight labels of up to eight display columns each.

To use soft labels, `slk_init()` must be called before calling `initscr(3XCURSES)`, `newterm(3XCURSES)`, or `ripoffline(3XCURSES)`. If `initscr()` eventually uses a line from *stdscr* to emulate the soft labels, then *fmt* determines how the labels are arranged on the screen. Setting *fmt* to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement. Other values for *fmt* are unspecified.

The `slk_init()` function has the effect of calling `ripoffline()` to reserve one screen line to accommodate the requested format.

The `slk_set()` and `slk_wset()` functions specify the text of soft label number *labnum*, within the range from 1 to and including 8. The *label* argument is the string to be put the lable. With `slk_set()` and `slk_wset()`, the width of the label is limited to eight columns positions. A null string or a null pointer specifies a blank label. The *justify* argument can have the following values to indicate how to justify *label* within the space reserved for it:

- 0 Align the start of *label* with the start of the space
- 1 Center *label* within the space
- 2 Align the end of *label* with the end of the space

The `slk_refresh()` and `slk_noutrefresh()` functions correspond to the `wrefresh(3XCURSES)` and `wnoutrefresh(3XCURSES)` functions.

The `slk_label()` function obtains soft label number *labnum*.

The `slk_clear()` function immediately clears the soft labels from the screen.

The `slk_restore()` function immediately restores the soft labels to the screen after a call to `slk_clear()`.

The `slk_touch()` function forces all the soft labels to be output the next time `slk_refresh()` or `slk_noutrefresh()` is called.

The `slk_attron()`, `slk_attrset()`, and `slk_attroff()` functions correspond to the `attron(3XCURSES)`, `attrset(3XCURSES)`, and `attroff(3XCURSES)` functions. They have an effect only if soft labels are stimulated on the bottom line of the screen.

The `slk_attr_on()`, `slk_attr_off()`, `slk_attr_set()` and `slk_color()` functions correspond to the `attr_on(3XCURSES)`, `attr_off(3XCURSES)`, `attr_set(3XCURSES)`, and `color_set(3XCURSES)` functions. As a result, they support color and the attribute constants with the `WA_` prefix.

slk_init(3XCURSES)

The *opts* argument is reserved for definition in a future release. Currently, the *opts* argument is a null pointer.

RETURN VALUES Upon successful completion, the `slk_label()` function returns the requested label with leading and trailing blanks stripped. Otherwise, it returns a null pointer.

Upon successful completion, the other functions return OK. Otherwise, they return ERR.

ERRORS No errors are defined.

USAGE When using multi-byte character sets, applications should check the width of the string by calling `mbstowcs(3C)` and then `wcswidth(3C)` before calling `slk_set()`. When using wide characters, applications should check the width of the string by calling `wcswidth()` before calling `slk_set()`.

Since the number of columns that a wide string will occupy is codeset-specific, call `wcwidth(3C)` and `wcswidth(3C)` to check the number of column positions in the string before calling `slk_wset()`.

Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attr_get(3XCURSES)`, `attroff(3XCURSES)`, `delscreen(3XCURSES)`, `libcurses(3XCURSES)`, `mbstowcs(3C)`, `ripcoffline(3XCURSES)`, `wcswidth(3C)`, `wcwidth(3C)`, `attributes(5)`, `standards(5)`

slk_label(3CURSES)

| | |
|--------------------|--|
| NAME | <code> curs_slk, slk_init, slk_set, slk_refresh, slk_noutrefresh, slk_label, slk_clear, slk_restore, slk_touch, slk_attron, slk_attrset, slk_attroff</code> – curses soft label routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int slk_init(int fmt); int slk_set(int labnum, char *label, int fmt); int slk_refresh(void); int slk_noutrefresh(void); char *slk_label(int labnum); int slk_clear(void); int slk_restore(void); int slk_touch(void); int slk_attron(chtype attrs); int slk_attrset(chtype attrs); int slk_attroff(chtype attrs);</pre> |
| DESCRIPTION | <p>curses manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, <code>curses</code> takes over the bottom line of <code>stdscr</code>, reducing the size of <code>stdscr</code> and the variable <code>LINES</code>. <code>curses</code> standardizes on eight labels of up to eight characters each.</p> <p>To use soft labels, the <code>slk_init()</code> routine must be called before <code>initscr()</code> or <code>newterm()</code> is called. If <code>initscr()</code> eventually uses a line from <code>stdscr</code> to emulate the soft labels, then <code>fmt</code> determines how the labels are arranged on the screen. Setting <code>fmt</code> to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement.</p> <p>With the <code>slk_set()</code> routine, <code>labnum</code> is the label number, from 1 to 8. <code>label</code> is the string to be put on the label, up to eight characters in length. A null string or a null pointer sets up a blank label. <code>fmt</code> is either 0, 1, or 2, indicating whether the label is to be left-justified, centered, or right-justified, respectively, within the label.</p> <p>The <code>slk_refresh()</code> and <code>slk_noutrefresh()</code> routines correspond to the <code>wrefresh()</code> and <code>wnoutrefresh()</code> routines.</p> <p>With the <code>slk_label()</code> routine, the current label for label number <code>labnum</code> is returned with leading and trailing blanks stripped.</p> <p>With the <code>slk_clear()</code> routine, the soft labels are cleared from the screen.</p> <p>With the <code>slk_restore()</code> routine, the soft labels are restored to the screen after a <code>slk_clear()</code> is performed.</p> |

slk_label(3CURSES)

With the `slk_touch()` routine, all the soft labels are forced to be output the next time a `slk_noutrefresh()` is performed.

The `slk_attron()`, `slk_attrset()`, and `slk_attroff()` routines correspond to `attron()`, `attrset()`, and `attroff()`. They have an effect only if soft labels are simulated on the bottom line of the screen.

RETURN VALUES Routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`slk_label()` returns `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_attr(3CURSES)`, `curl_initscr(3CURSES)`, `curl_refresh(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.

slk_label(3XCURSES)

| | | | | | | | | | | | | | | | |
|--------------------------|--|--------------|---|-------------|---|--------------------------|------------------|------------|---|---------------|----------------------------------|--------------|--|----------------|---|
| NAME | slk_attroff, slk_attr_off, slk_attron, slk_attr_on, slk_attrset, slk_attr_set, slk_clear, slk_color, slk_init, slk_label, slk_noutrefresh, slk_refresh, slk_restore, slk_set, slk_touch, slk_wset – soft label functions | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int slk_attroff(const chtype attrs); int slk_attr_off(const attr_t attrs, void *opts); int slk_attron(const chtype attrs); int slk_attr_on(const attr_t attrs, void *opts); int slk_attrset(const chtype attrs); int slk_attr_set(const attr_t attrs, short color_pair_number, void *opts); int slk_clear(void); int slk_color(short color_pair_number); int slk_init(int fmt); char *slk_label(int labnum); int slk_noutrefresh(void); int slk_refresh(void); int slk_restore(void); int slk_set(int labnum, const char *label, int justify); int slk_touch(void); int slk_wset(int labnum, const wchar_t *label, int justify);</pre> | | | | | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>attrs</i></td><td>are the window attributes to be added or removed.</td></tr><tr><td><i>opts</i></td><td>Is reserved for future use. Currently, this must be a null pointer.</td></tr><tr><td><i>color_pair_number</i></td><td>Is a color pair.</td></tr><tr><td><i>fmt</i></td><td>Is the format of how the labels are arranged on the screen.</td></tr><tr><td><i>labnum</i></td><td>Is the number of the soft label.</td></tr><tr><td><i>label</i></td><td>Is the name to be given to a soft label.</td></tr><tr><td><i>justify</i></td><td>Is a number indicating how to justify the label name.</td></tr></table> | <i>attrs</i> | are the window attributes to be added or removed. | <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | <i>color_pair_number</i> | Is a color pair. | <i>fmt</i> | Is the format of how the labels are arranged on the screen. | <i>labnum</i> | Is the number of the soft label. | <i>label</i> | Is the name to be given to a soft label. | <i>justify</i> | Is a number indicating how to justify the label name. |
| <i>attrs</i> | are the window attributes to be added or removed. | | | | | | | | | | | | | | |
| <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | | | | | | | | | | | | | | |
| <i>color_pair_number</i> | Is a color pair. | | | | | | | | | | | | | | |
| <i>fmt</i> | Is the format of how the labels are arranged on the screen. | | | | | | | | | | | | | | |
| <i>labnum</i> | Is the number of the soft label. | | | | | | | | | | | | | | |
| <i>label</i> | Is the name to be given to a soft label. | | | | | | | | | | | | | | |
| <i>justify</i> | Is a number indicating how to justify the label name. | | | | | | | | | | | | | | |

DESCRIPTION | The Curses interface manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, Curses takes over the bottom line of *stdscr*, reducing the size of *stdscr* and the value of the `LINES` external variable. There can be up to eight labels of up to eight display columns each.

To use soft labels, `slk_init()` must be called before calling `initscr(3XCURSES)`, `newterm(3XCURSES)`, or `ripoffline(3XCURSES)`. If `initscr()` eventually uses a line from *stdscr* to emulate the soft labels, then *fmt* determines how the labels are arranged on the screen. Setting *fmt* to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement. Other values for *fmt* are unspecified.

The `slk_init()` function has the effect of calling `ripoffline()` to reserve one screen line to accommodate the requested format.

The `slk_set()` and `slk_wset()` functions specify the text of soft label number *labnum*, within the range from 1 to and including 8. The *label* argument is the string to be put the lable. With `slk_set()` and `slk_wset()`, the width of the label is limited to eight columns positions. A null string or a null pointer specifies a blank label. The *justify* argument can have the following values to indicate how to justify *label* within the space reserved for it:

- 0 Align the start of *label* with the start of the space
- 1 Center *label* within the space
- 2 Align the end of *label* with the end of the space

The `slk_refresh()` and `slk_noutrefresh()` functions correspond to the `wrefresh(3XCURSES)` and `wnoutrefresh(3XCURSES)` functions.

The `slk_label()` function obtains soft label number *labnum*.

The `slk_clear()` function immediately clears the soft labels from the screen.

The `slk_restore()` function immediately restores the soft labels to the screen after a call to `slk_clear()`.

The `slk_touch()` function forces all the soft labels to be output the next time `slk_refresh()` or `slk_noutrefresh()` is called.

The `slk_attron()`, `slk_attrset()`, and `slk_attroff()` functions correspond to the `attron(3XCURSES)`, `attrset(3XCURSES)`, and `attroff(3XCURSES)` functions. They have an effect only if soft labels are stimulated on the bottom line of the screen.

The `slk_attr_on()`, `slk_attr_off()`, `slk_attr_set()` and `slk_color()` functions correspond to the `attr_on(3XCURSES)`, `attr_off(3XCURSES)`, `attr_set(3XCURSES)`, and `color_set(3XCURSES)` functions. As a result, they support color and the attribute constants with the `WA_` prefix.

slk_label(3XCURSES)

The *opts* argument is reserved for definition in a future release. Currently, the *opts* argument is a null pointer.

RETURN VALUES Upon successful completion, the `slk_label()` function returns the requested label with leading and trailing blanks stripped. Otherwise, it returns a null pointer.

Upon successful completion, the other functions return OK. Otherwise, they return ERR.

ERRORS No errors are defined.

USAGE When using multi-byte character sets, applications should check the width of the string by calling `mbstowcs(3C)` and then `wcswidth(3C)` before calling `slk_set()`. When using wide characters, applications should check the width of the string by calling `wcswidth()` before calling `slk_set()`.

Since the number of columns that a wide string will occupy is codeset-specific, call `wcwidth(3C)` and `wcswidth(3C)` to check the number of column positions in the string before calling `slk_wset()`.

Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attr_get(3XCURSES)`, `attroff(3XCURSES)`, `delscreen(3XCURSES)`, `libcurses(3XCURSES)`, `mbstowcs(3C)`, `ripoffline(3XCURSES)`, `wcswidth(3C)`, `wcwidth(3C)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | curs_slk, slk_init, slk_set, slk_refresh, slk_noutrefresh, slk_label, slk_clear, slk_restore, slk_touch, slk_atron, slk_attrset, slk_attroff – curses soft label routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int slk_init(int <i>fmt</i>); int slk_set(int <i>labnum</i>, char *<i>label</i>, int <i>fmt</i>); int slk_refresh(void); int slk_noutrefresh(void); char *slk_label(int <i>labnum</i>); int slk_clear(void); int slk_restore(void); int slk_touch(void); int slk_atron(chtype <i>attrs</i>); int slk_attrset(chtype <i>attrs</i>); int slk_attroff(chtype <i>attrs</i>);</pre> |
| DESCRIPTION | <p>curses manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, <i>curses</i> takes over the bottom line of <code>stdscr</code>, reducing the size of <code>stdscr</code> and the variable <code>LINES</code>. <i>curses</i> standardizes on eight labels of up to eight characters each.</p> <p>To use soft labels, the <code>slk_init()</code> routine must be called before <code>initscr()</code> or <code>newterm()</code> is called. If <code>initscr()</code> eventually uses a line from <code>stdscr</code> to emulate the soft labels, then <code>fmt</code> determines how the labels are arranged on the screen. Setting <code>fmt</code> to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement.</p> <p>With the <code>slk_set()</code> routine, <i>labnum</i> is the label number, from 1 to 8. <i>label</i> is the string to be put on the label, up to eight characters in length. A null string or a null pointer sets up a blank label. <code>fmt</code> is either 0, 1, or 2, indicating whether the label is to be left-justified, centered, or right-justified, respectively, within the label.</p> <p>The <code>slk_refresh()</code> and <code>slk_noutrefresh()</code> routines correspond to the <code>wrefresh()</code> and <code>wnoutrefresh()</code> routines.</p> <p>With the <code>slk_label()</code> routine, the current label for label number <i>labnum</i> is returned with leading and trailing blanks stripped.</p> <p>With the <code>slk_clear()</code> routine, the soft labels are cleared from the screen.</p> <p>With the <code>slk_restore()</code> routine, the soft labels are restored to the screen after a <code>slk_clear()</code> is performed.</p> |

slk_noutrefresh(3CURSES)

With the `slk_touch()` routine, all the soft labels are forced to be output the next time a `slk_noutrefresh()` is performed.

The `slk_attron()`, `slk_attrset()`, and `slk_attroff()` routines correspond to `attron()`, `attrset()`, and `attroff()`. They have an effect only if soft labels are simulated on the bottom line of the screen.

RETURN VALUES Routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`slk_label()` returns `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_attr(3CURSES)`, `curl_initscr(3CURSES)`, `curl_refresh(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.

| | | | | | | | | | | | | | | | |
|--------------------------|--|--------------|---|-------------|---|--------------------------|------------------|------------|---|---------------|----------------------------------|--------------|--|----------------|---|
| NAME | slk_attroff, slk_attr_off, slk_attron, slk_attr_on, slk_attrset, slk_attr_set, slk_clear, slk_color, slk_init, slk_label, slk_noutrefresh, slk_refresh, slk_restore, slk_set, slk_touch, slk_wset – soft label functions | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int slk_attroff(const chtype attrs); int slk_attr_off(const attr_t attrs, void *opts); int slk_attron(const chtype attrs); int slk_attr_on(const attr_t attrs, void *opts); int slk_attrset(const chtype attrs); int slk_attr_set(const attr_t attrs, short color_pair_number, void *opts); int slk_clear(void); int slk_color(short color_pair_number); int slk_init(int fmt); char *slk_label(int labnum); int slk_noutrefresh(void); int slk_refresh(void); int slk_restore(void); int slk_set(int labnum, const char *label, int justify); int slk_touch(void); int slk_wset(int labnum, const wchar_t *label, int justify);</pre> | | | | | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>attrs</i></td> <td>are the window attributes to be added or removed.</td> </tr> <tr> <td><i>opts</i></td> <td>Is reserved for future use. Currently, this must be a null pointer.</td> </tr> <tr> <td><i>color_pair_number</i></td> <td>Is a color pair.</td> </tr> <tr> <td><i>fmt</i></td> <td>Is the format of how the labels are arranged on the screen.</td> </tr> <tr> <td><i>labnum</i></td> <td>Is the number of the soft label.</td> </tr> <tr> <td><i>label</i></td> <td>Is the name to be given to a soft label.</td> </tr> <tr> <td><i>justify</i></td> <td>Is a number indicating how to justify the label name.</td> </tr> </table> | <i>attrs</i> | are the window attributes to be added or removed. | <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | <i>color_pair_number</i> | Is a color pair. | <i>fmt</i> | Is the format of how the labels are arranged on the screen. | <i>labnum</i> | Is the number of the soft label. | <i>label</i> | Is the name to be given to a soft label. | <i>justify</i> | Is a number indicating how to justify the label name. |
| <i>attrs</i> | are the window attributes to be added or removed. | | | | | | | | | | | | | | |
| <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | | | | | | | | | | | | | | |
| <i>color_pair_number</i> | Is a color pair. | | | | | | | | | | | | | | |
| <i>fmt</i> | Is the format of how the labels are arranged on the screen. | | | | | | | | | | | | | | |
| <i>labnum</i> | Is the number of the soft label. | | | | | | | | | | | | | | |
| <i>label</i> | Is the name to be given to a soft label. | | | | | | | | | | | | | | |
| <i>justify</i> | Is a number indicating how to justify the label name. | | | | | | | | | | | | | | |

slk_noutrefresh(3XCURSES)

DESCRIPTION

The Curses interface manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, Curses takes over the bottom line of *stdscr*, reducing the size of *stdscr* and the value of the `LINES` external variable. There can be up to eight labels of up to eight display columns each.

To use soft labels, `slk_init()` must be called before calling `initscr(3XCURSES)`, `newterm(3XCURSES)`, or `ripoffline(3XCURSES)`. If `initscr()` eventually uses a line from *stdscr* to emulate the soft labels, then *fmt* determines how the labels are arranged on the screen. Setting *fmt* to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement. Other values for *fmt* are unspecified.

The `slk_init()` function has the effect of calling `ripoffline()` to reserve one screen line to accommodate the requested format.

The `slk_set()` and `slk_wset()` functions specify the text of soft label number *labnum*, within the range from 1 to and including 8. The *label* argument is the string to be put the lable. With `slk_set()` and `slk_wset()`, the width of the label is limited to eight columns positions. A null string or a null pointer specifies a blank label. The *justify* argument can have the following values to indicate how to justify *label* within the space reserved for it:

- 0 Align the start of *label* with the start of the space
- 1 Center *label* within the space
- 2 Align the end of *label* with the end of the space

The `slk_refresh()` and `slk_noutrefresh()` functions correspond to the `wrefresh(3XCURSES)` and `wnoutrefresh(3XCURSES)` functions.

The `slk_label()` function obtains soft label number *labnum*.

The `slk_clear()` function immediately clears the soft labels from the screen.

The `slk_restore()` function immediately restores the soft labels to the screen after a call to `slk_clear()`.

The `slk_touch()` function forces all the soft labels to be output the next time `slk_refresh()` or `slk_noutrefresh()` is called.

The `slk_attron()`, `slk_attrset()`, and `slk_attroff()` functions correspond to the `attron(3XCURSES)`, `attrset(3XCURSES)`, and `attroff(3XCURSES)` functions. They have an effect only if soft labels are stimulated on the bottom line of the screen.

The `slk_attr_on()`, `slk_attr_off()`, `slk_attr_set()` and `slk_color()` functions correspond to the `attr_on(3XCURSES)`, `attr_off(3XCURSES)`, `attr_set(3XCURSES)`, and `color_set(3XCURSES)` functions. As a result, they support color and the attribute constants with the `WA_` prefix.

slk_noutrefresh(3XCURSES)

The *opts* argument is reserved for definition in a future release. Currently, the *opts* argument is a null pointer.

RETURN VALUES Upon successful completion, the `slk_label()` function returns the requested label with leading and trailing blanks stripped. Otherwise, it returns a null pointer.

Upon successful completion, the other functions return OK. Otherwise, they return ERR.

ERRORS No errors are defined.

USAGE When using multi-byte character sets, applications should check the width of the string by calling `mbstowcs(3C)` and then `wcswidth(3C)` before calling `slk_set()`. When using wide characters, applications should check the width of the string by calling `wcswidth()` before calling `slk_set()`.

Since the number of columns that a wide string will occupy is codeset-specific, call `wcwidth(3C)` and `wcswidth(3C)` to check the number of column positions in the string before calling `slk_wset()`.

Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attr_get(3XCURSES)`, `attroff(3XCURSES)`, `delscreen(3XCURSES)`, `libcurses(3XCURSES)`, `mbstowcs(3C)`, `ripcoffline(3XCURSES)`, `wcswidth(3C)`, `wcwidth(3C)`, `attributes(5)`, `standards(5)`

slk_refresh(3CURSES)

| | |
|--------------------|--|
| NAME | <code> curs_slk, slk_init, slk_set, slk_refresh, slk_noutrefresh, slk_label, slk_clear, slk_restore, slk_touch, slk_attron, slk_attrset, slk_attroff</code> – curses soft label routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int slk_init(int <i>fmt</i>); int slk_set(int <i>labnum</i>, char *<i>label</i>, int <i>fmt</i>); int slk_refresh(void); int slk_noutrefresh(void); char *slk_label(int <i>labnum</i>); int slk_clear(void); int slk_restore(void); int slk_touch(void); int slk_attron(chtype <i>attrs</i>); int slk_attrset(chtype <i>attrs</i>); int slk_attroff(chtype <i>attrs</i>);</pre> |
| DESCRIPTION | <p>curses manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, <code>curses</code> takes over the bottom line of <code>stdscr</code>, reducing the size of <code>stdscr</code> and the variable <code>LINES</code>. <code>curses</code> standardizes on eight labels of up to eight characters each.</p> <p>To use soft labels, the <code>slk_init()</code> routine must be called before <code>initscr()</code> or <code>newterm()</code> is called. If <code>initscr()</code> eventually uses a line from <code>stdscr</code> to emulate the soft labels, then <code>fmt</code> determines how the labels are arranged on the screen. Setting <code>fmt</code> to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement.</p> <p>With the <code>slk_set()</code> routine, <i>labnum</i> is the label number, from 1 to 8. <i>label</i> is the string to be put on the label, up to eight characters in length. A null string or a null pointer sets up a blank label. <code>fmt</code> is either 0, 1, or 2, indicating whether the label is to be left-justified, centered, or right-justified, respectively, within the label.</p> <p>The <code>slk_refresh()</code> and <code>slk_noutrefresh()</code> routines correspond to the <code>wrefresh()</code> and <code>wnoutrefresh()</code> routines.</p> <p>With the <code>slk_label()</code> routine, the current label for label number <i>labnum</i> is returned with leading and trailing blanks stripped.</p> <p>With the <code>slk_clear()</code> routine, the soft labels are cleared from the screen.</p> <p>With the <code>slk_restore()</code> routine, the soft labels are restored to the screen after a <code>slk_clear()</code> is performed.</p> |

slk_refresh(3CURSES)

With the `slk_touch()` routine, all the soft labels are forced to be output the next time a `slk_noutrefresh()` is performed.

The `slk_attron()`, `slk_attrset()`, and `slk_attroff()` routines correspond to `attron()`, `attrset()`, and `attroff()`. They have an effect only if soft labels are simulated on the bottom line of the screen.

RETURN VALUES Routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`slk_label()` returns `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_attr(3CURSES)`, `curl_initscr(3CURSES)`, `curl_refresh(3CURSES)`, `curls(3CURSES)`, `attributes(5)`

NOTES The header `< curses.h >` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.

slk_refresh(3XCURSES)

| | | | | | | | | | | | | | | | |
|--------------------------|--|--------------|---|-------------|---|--------------------------|------------------|------------|---|---------------|----------------------------------|--------------|--|----------------|---|
| NAME | slk_attroff, slk_attr_off, slk_attron, slk_attr_on, slk_attrset, slk_attr_set, slk_clear, slk_color, slk_init, slk_label, slk_noutrefresh, slk_refresh, slk_restore, slk_set, slk_touch, slk_wset – soft label functions | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int slk_attroff(const chtype attrs); int slk_attr_off(const attr_t attrs, void *opts); int slk_attron(const chtype attrs); int slk_attr_on(const attr_t attrs, void *opts); int slk_attrset(const chtype attrs); int slk_attr_set(const attr_t attrs, short color_pair_number, void *opts); int slk_clear(void); int slk_color(short color_pair_number); int slk_init(int fmt); char *slk_label(int labnum); int slk_noutrefresh(void); int slk_refresh(void); int slk_restore(void); int slk_set(int labnum, const char *label, int justify); int slk_touch(void); int slk_wset(int labnum, const wchar_t *label, int justify);</pre> | | | | | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>attrs</i></td><td>are the window attributes to be added or removed.</td></tr><tr><td><i>opts</i></td><td>Is reserved for future use. Currently, this must be a null pointer.</td></tr><tr><td><i>color_pair_number</i></td><td>Is a color pair.</td></tr><tr><td><i>fmt</i></td><td>Is the format of how the labels are arranged on the screen.</td></tr><tr><td><i>labnum</i></td><td>Is the number of the soft label.</td></tr><tr><td><i>label</i></td><td>Is the name to be given to a soft label.</td></tr><tr><td><i>justify</i></td><td>Is a number indicating how to justify the label name.</td></tr></table> | <i>attrs</i> | are the window attributes to be added or removed. | <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | <i>color_pair_number</i> | Is a color pair. | <i>fmt</i> | Is the format of how the labels are arranged on the screen. | <i>labnum</i> | Is the number of the soft label. | <i>label</i> | Is the name to be given to a soft label. | <i>justify</i> | Is a number indicating how to justify the label name. |
| <i>attrs</i> | are the window attributes to be added or removed. | | | | | | | | | | | | | | |
| <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | | | | | | | | | | | | | | |
| <i>color_pair_number</i> | Is a color pair. | | | | | | | | | | | | | | |
| <i>fmt</i> | Is the format of how the labels are arranged on the screen. | | | | | | | | | | | | | | |
| <i>labnum</i> | Is the number of the soft label. | | | | | | | | | | | | | | |
| <i>label</i> | Is the name to be given to a soft label. | | | | | | | | | | | | | | |
| <i>justify</i> | Is a number indicating how to justify the label name. | | | | | | | | | | | | | | |

DESCRIPTION

The Curses interface manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, Curses takes over the bottom line of *stdscr*, reducing the size of *stdscr* and the value of the `LINES` external variable. There can be up to eight labels of up to eight display columns each.

To use soft labels, `slk_init()` must be called before calling `initscr(3XCURSES)`, `newterm(3XCURSES)`, or `ripoffline(3XCURSES)`. If `initscr()` eventually uses a line from *stdscr* to emulate the soft labels, then *fmt* determines how the labels are arranged on the screen. Setting *fmt* to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement. Other values for *fmt* are unspecified.

The `slk_init()` function has the effect of calling `ripoffline()` to reserve one screen line to accommodate the requested format.

The `slk_set()` and `slk_wset()` functions specify the text of soft label number *labnum*, within the range from 1 to and including 8. The *label* argument is the string to be put the lable. With `slk_set()` and `slk_wset()`, the width of the label is limited to eight columns positions. A null string or a null pointer specifies a blank label. The *justify* argument can have the following values to indicate how to justify *label* within the space reserved for it:

- 0 Align the start of *label* with the start of the space
- 1 Center *label* within the space
- 2 Align the end of *label* with the end of the space

The `slk_refresh()` and `slk_noutrefresh()` functions correspond to the `wrefresh(3XCURSES)` and `wnoutrefresh(3XCURSES)` functions.

The `slk_label()` function obtains soft label number *labnum*.

The `slk_clear()` function immediately clears the soft labels from the screen.

The `slk_restore()` function immediately restores the soft labels to the screen after a call to `slk_clear()`.

The `slk_touch()` function forces all the soft labels to be output the next time `slk_refresh()` or `slk_noutrefresh()` is called.

The `slk_attron()`, `slk_attrset()`, and `slk_attroff()` functions correspond to the `attron(3XCURSES)`, `attrset(3XCURSES)`, and `attroff(3XCURSES)` functions. They have an effect only if soft labels are stimulated on the bottom line of the screen.

The `slk_attr_on()`, `slk_attr_off()`, `slk_attr_set()` and `slk_color()` functions correspond to the `attr_on(3XCURSES)`, `attr_off(3XCURSES)`, `attr_set(3XCURSES)`, and `color_set(3XCURSES)` functions. As a result, they support color and the attribute constants with the `WA_` prefix.

slk_refresh(3XCURSES)

- The *opts* argument is reserved for definition in a future release. Currently, the *opts* argument is a null pointer.
- RETURN VALUES** Upon successful completion, the `slk_label()` function returns the requested label with leading and trailing blanks stripped. Otherwise, it returns a null pointer.
- Upon successful completion, the other functions return OK. Otherwise, they return ERR.
- ERRORS** No errors are defined.
- USAGE** When using multi-byte character sets, applications should check the width of the string by calling `mbstowcs(3C)` and then `wcswidth(3C)` before calling `slk_set()`. When using wide characters, applications should check the width of the string by calling `wcswidth()` before calling `slk_set()`.
- Since the number of columns that a wide string will occupy is codeset-specific, call `wcwidth(3C)` and `wcswidth(3C)` to check the number of column positions in the string before calling `slk_wset()`.
- Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.
- ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

- SEE ALSO** `attr_get(3XCURSES)`, `attroff(3XCURSES)`, `delscreen(3XCURSES)`, `libcurses(3XCURSES)`, `mbstowcs(3C)`, `ripoffline(3XCURSES)`, `wcswidth(3C)`, `wcwidth(3C)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | curs_slk, slk_init, slk_set, slk_refresh, slk_noutrefresh, slk_label, slk_clear, slk_restore, slk_touch, slk_attron, slk_attrset, slk_attroff – curses soft label routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int slk_init(int fmt); int slk_set(int labnum, char *label, int fmt); int slk_refresh(void); int slk_noutrefresh(void); char *slk_label(int labnum); int slk_clear(void); int slk_restore(void); int slk_touch(void); int slk_attron(chtype attrs); int slk_attrset(chtype attrs); int slk_attroff(chtype attrs);</pre> |
| DESCRIPTION | <p>curses manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, <i>curses</i> takes over the bottom line of <code>stdscr</code>, reducing the size of <code>stdscr</code> and the variable <code>LINES</code>. <i>curses</i> standardizes on eight labels of up to eight characters each.</p> <p>To use soft labels, the <code>slk_init()</code> routine must be called before <code>initscr()</code> or <code>newterm()</code> is called. If <code>initscr()</code> eventually uses a line from <code>stdscr</code> to emulate the soft labels, then <code>fmt</code> determines how the labels are arranged on the screen. Setting <code>fmt</code> to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement.</p> <p>With the <code>slk_set()</code> routine, <i>labnum</i> is the label number, from 1 to 8. <i>label</i> is the string to be put on the label, up to eight characters in length. A null string or a null pointer sets up a blank label. <code>fmt</code> is either 0, 1, or 2, indicating whether the label is to be left-justified, centered, or right-justified, respectively, within the label.</p> <p>The <code>slk_refresh()</code> and <code>slk_noutrefresh()</code> routines correspond to the <code>wrefresh()</code> and <code>wnoutrefresh()</code> routines.</p> <p>With the <code>slk_label()</code> routine, the current label for label number <i>labnum</i> is returned with leading and trailing blanks stripped.</p> <p>With the <code>slk_clear()</code> routine, the soft labels are cleared from the screen.</p> <p>With the <code>slk_restore()</code> routine, the soft labels are restored to the screen after a <code>slk_clear()</code> is performed.</p> |

slk_restore(3CURSES)

With the `slk_touch()` routine, all the soft labels are forced to be output the next time a `slk_noutrefresh()` is performed.

The `slk_attron()`, `slk_attrset()`, and `slk_attroff()` routines correspond to `attron()`, `attrset()`, and `attroff()`. They have an effect only if soft labels are simulated on the bottom line of the screen.

RETURN VALUES Routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`slk_label()` returns `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_attr(3CURSES)`, `curl_initscr(3CURSES)`, `curl_refresh(3CURSES)`, `curls(3CURSES)`, `attributes(5)`

NOTES The header `< curses.h >` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.

| | |
|-------------------|--|
| NAME | slk_attroff, slk_attr_off, slk_attron, slk_attr_on, slk_attrset, slk_attr_set, slk_clear, slk_color, slk_init, slk_label, slk_noutrefresh, slk_refresh, slk_restore, slk_set, slk_touch, slk_wset – soft label functions |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int slk_attroff(const chtype attrs); int slk_attr_off(const attr_t attrs, void *opts); int slk_attron(const chtype attrs); int slk_attr_on(const attr_t attrs, void *opts); int slk_attrset(const chtype attrs); int slk_attr_set(const attr_t attrs, short color_pair_number, void *opts); int slk_clear(void); int slk_color(short color_pair_number); int slk_init(int fmt); char *slk_label(int labnum); int slk_noutrefresh(void); int slk_refresh(void); int slk_restore(void); int slk_set(int labnum, const char *label, int justify); int slk_touch(void); int slk_wset(int labnum, const wchar_t *label, int justify);</pre> |
| PARAMETERS | <p><i>attrs</i> are the window attributes to be added or removed.</p> <p><i>opts</i> Is reserved for future use. Currently, this must be a null pointer.</p> <p><i>color_pair_number</i> Is a color pair.</p> <p><i>fmt</i> Is the format of how the labels are arranged on the screen.</p> <p><i>labnum</i> Is the number of the soft label.</p> <p><i>label</i> Is the name to be given to a soft label.</p> <p><i>justify</i> Is a number indicating how to justify the label name.</p> |

slk_restore(3XCURSES)

DESCRIPTION

The Curses interface manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, Curses takes over the bottom line of *stdscr*, reducing the size of *stdscr* and the value of the `LINES` external variable. There can be up to eight labels of up to eight display columns each.

To use soft labels, `slk_init()` must be called before calling `initscr(3XCURSES)`, `newterm(3XCURSES)`, or `ripoffline(3XCURSES)`. If `initscr()` eventually uses a line from *stdscr* to emulate the soft labels, then *fmt* determines how the labels are arranged on the screen. Setting *fmt* to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement. Other values for *fmt* are unspecified.

The `slk_init()` function has the effect of calling `ripoffline()` to reserve one screen line to accommodate the requested format.

The `slk_set()` and `slk_wset()` functions specify the text of soft label number *labnum*, within the range from 1 to and including 8. The *label* argument is the string to be put the lable. With `slk_set()` and `slk_wset()`, the width of the label is limited to eight columns positions. A null string or a null pointer specifies a blank label. The *justify* argument can have the following values to indicate how to justify *label* within the space reserved for it:

- 0 Align the start of *label* with the start of the space
- 1 Center *label* within the space
- 2 Align the end of *label* with the end of the space

The `slk_refresh()` and `slk_noutrefresh()` functions correspond to the `wrefresh(3XCURSES)` and `wnoutrefresh(3XCURSES)` functions.

The `slk_label()` function obtains soft label number *labnum*.

The `slk_clear()` function immediately clears the soft labels from the screen.

The `slk_restore()` function immediately restores the soft labels to the screen after a call to `slk_clear()`.

The `slk_touch()` function forces all the soft labels to be output the next time `slk_refresh()` or `slk_noutrefresh()` is called.

The `slk_attron()`, `slk_attrset()`, and `slk_attroff()` functions correspond to the `attron(3XCURSES)`, `attrset(3XCURSES)`, and `attroff(3XCURSES)` functions. They have an effect only if soft labels are stimulated on the bottom line of the screen.

The `slk_attr_on()`, `slk_attr_off()`, `slk_attr_set()` and `slk_color()` functions correspond to the `attr_on(3XCURSES)`, `attr_off(3XCURSES)`, `attr_set(3XCURSES)`, and `color_set(3XCURSES)` functions. As a result, they support color and the attribute constants with the `WA_` prefix.

slk_restore(3XCURSES)

The *opts* argument is reserved for definition in a future release. Currently, the *opts* argument is a null pointer.

RETURN VALUES Upon successful completion, the `slk_label()` function returns the requested label with leading and trailing blanks stripped. Otherwise, it returns a null pointer.

Upon successful completion, the other functions return OK. Otherwise, they return ERR.

ERRORS No errors are defined.

USAGE When using multi-byte character sets, applications should check the width of the string by calling `mbstowcs(3C)` and then `wcswidth(3C)` before calling `slk_set()`. When using wide characters, applications should check the width of the string by calling `wcswidth()` before calling `slk_set()`.

Since the number of columns that a wide string will occupy is codeset-specific, call `wcwidth(3C)` and `wcswidth(3C)` to check the number of column positions in the string before calling `slk_wset()`.

Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attr_get(3XCURSES)`, `attroff(3XCURSES)`, `delscreen(3XCURSES)`, `libcurses(3XCURSES)`, `mbstowcs(3C)`, `ripcoffline(3XCURSES)`, `wcswidth(3C)`, `wcwidth(3C)`, `attributes(5)`, `standards(5)`

slk_set(3CURSES)

| | |
|--------------------|--|
| NAME | <code> curs_slk, slk_init, slk_set, slk_refresh, slk_noutrefresh, slk_label, slk_clear, slk_restore, slk_touch, slk_attron, slk_attrset, slk_attroff</code> – curses soft label routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int slk_init(int fmt); int slk_set(int labnum, char *label, int fmt); int slk_refresh(void); int slk_noutrefresh(void); char *slk_label(int labnum); int slk_clear(void); int slk_restore(void); int slk_touch(void); int slk_attron(chtype attrs); int slk_attrset(chtype attrs); int slk_attroff(chtype attrs);</pre> |
| DESCRIPTION | <p>curses manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, <code>curses</code> takes over the bottom line of <code>stdscr</code>, reducing the size of <code>stdscr</code> and the variable <code>LINES</code>. <code>curses</code> standardizes on eight labels of up to eight characters each.</p> <p>To use soft labels, the <code>slk_init()</code> routine must be called before <code>initscr()</code> or <code>newterm()</code> is called. If <code>initscr()</code> eventually uses a line from <code>stdscr</code> to emulate the soft labels, then <code>fmt</code> determines how the labels are arranged on the screen. Setting <code>fmt</code> to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement.</p> <p>With the <code>slk_set()</code> routine, <code>labnum</code> is the label number, from 1 to 8. <code>label</code> is the string to be put on the label, up to eight characters in length. A null string or a null pointer sets up a blank label. <code>fmt</code> is either 0, 1, or 2, indicating whether the label is to be left-justified, centered, or right-justified, respectively, within the label.</p> <p>The <code>slk_refresh()</code> and <code>slk_noutrefresh()</code> routines correspond to the <code>wrefresh()</code> and <code>wnoutrefresh()</code> routines.</p> <p>With the <code>slk_label()</code> routine, the current label for label number <code>labnum</code> is returned with leading and trailing blanks stripped.</p> <p>With the <code>slk_clear()</code> routine, the soft labels are cleared from the screen.</p> <p>With the <code>slk_restore()</code> routine, the soft labels are restored to the screen after a <code>slk_clear()</code> is performed.</p> |

slk_set(3CURSES)

With the `slk_touch()` routine, all the soft labels are forced to be output the next time a `slk_noutrefresh()` is performed.

The `slk_attron()`, `slk_attrset()`, and `slk_attroff()` routines correspond to `attron()`, `attrset()`, and `attroff()`. They have an effect only if soft labels are simulated on the bottom line of the screen.

RETURN VALUES Routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`slk_label()` returns `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_attr(3CURSES)`, `curl_initscr(3CURSES)`, `curl_refresh(3CURSES)`, `curls(3CURSES)`, `attributes(5)`

NOTES The header `< curses.h >` automatically includes the headers `<stdio.h >` and `<unctrl.h >`.

Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.

slk_set(3XCURSES)

| | | | | | | | | | | | | | | | |
|--------------------------|---|--------------|---|-------------|---|--------------------------|------------------|------------|---|---------------|----------------------------------|--------------|--|----------------|---|
| NAME | slk_atroff, slk_attr_off, slk_attron, slk_attr_on, slk_attrset, slk_attr_set, slk_clear, slk_color, slk_init, slk_label, slk_noutrefresh, slk_refresh, slk_restore, slk_set, slk_touch, slk_wset – soft label functions | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int slk_atroff(const chtype attrs); int slk_attr_off(const attr_t attrs, void *opts); int slk_attron(const chtype attrs); int slk_attr_on(const attr_t attrs, void *opts); int slk_attrset(const chtype attrs); int slk_attr_set(const attr_t attrs, short color_pair_number, void *opts); int slk_clear(void); int slk_color(short color_pair_number); int slk_init(int fmt); char *slk_label(int labnum); int slk_noutrefresh(void); int slk_refresh(void); int slk_restore(void); int slk_set(int labnum, const char *label, int justify); int slk_touch(void); int slk_wset(int labnum, const wchar_t *label, int justify);</pre> | | | | | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>attrs</i></td><td>are the window attributes to be added or removed.</td></tr><tr><td><i>opts</i></td><td>Is reserved for future use. Currently, this must be a null pointer.</td></tr><tr><td><i>color_pair_number</i></td><td>Is a color pair.</td></tr><tr><td><i>fmt</i></td><td>Is the format of how the labels are arranged on the screen.</td></tr><tr><td><i>labnum</i></td><td>Is the number of the soft label.</td></tr><tr><td><i>label</i></td><td>Is the name to be given to a soft label.</td></tr><tr><td><i>justify</i></td><td>Is a number indicating how to justify the label name.</td></tr></table> | <i>attrs</i> | are the window attributes to be added or removed. | <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | <i>color_pair_number</i> | Is a color pair. | <i>fmt</i> | Is the format of how the labels are arranged on the screen. | <i>labnum</i> | Is the number of the soft label. | <i>label</i> | Is the name to be given to a soft label. | <i>justify</i> | Is a number indicating how to justify the label name. |
| <i>attrs</i> | are the window attributes to be added or removed. | | | | | | | | | | | | | | |
| <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | | | | | | | | | | | | | | |
| <i>color_pair_number</i> | Is a color pair. | | | | | | | | | | | | | | |
| <i>fmt</i> | Is the format of how the labels are arranged on the screen. | | | | | | | | | | | | | | |
| <i>labnum</i> | Is the number of the soft label. | | | | | | | | | | | | | | |
| <i>label</i> | Is the name to be given to a soft label. | | | | | | | | | | | | | | |
| <i>justify</i> | Is a number indicating how to justify the label name. | | | | | | | | | | | | | | |

DESCRIPTION

The Curses interface manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, Curses takes over the bottom line of *stdscr*, reducing the size of *stdscr* and the value of the `LINES` external variable. There can be up to eight labels of up to eight display columns each.

To use soft labels, `slk_init()` must be called before calling `initscr(3XCURSES)`, `newterm(3XCURSES)`, or `ripoffline(3XCURSES)`. If `initscr()` eventually uses a line from *stdscr* to emulate the soft labels, then *fmt* determines how the labels are arranged on the screen. Setting *fmt* to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement. Other values for *fmt* are unspecified.

The `slk_init()` function has the effect of calling `ripoffline()` to reserve one screen line to accommodate the requested format.

The `slk_set()` and `slk_wset()` functions specify the text of soft label number *labnum*, within the range from 1 to and including 8. The *label* argument is the string to be put the lable. With `slk_set()` and `slk_wset()`, the width of the label is limited to eight columns positions. A null string or a null pointer specifies a blank label. The *justify* argument can have the following values to indicate how to justify *label* within the space reserved for it:

- 0 Align the start of *label* with the start of the space
- 1 Center *label* within the space
- 2 Align the end of *label* with the end of the space

The `slk_refresh()` and `slk_noutrefresh()` functions correspond to the `wrefresh(3XCURSES)` and `wnoutrefresh(3XCURSES)` functions.

The `slk_label()` function obtains soft label number *labnum*.

The `slk_clear()` function immediately clears the soft labels from the screen.

The `slk_restore()` function immediately restores the soft labels to the screen after a call to `slk_clear()`.

The `slk_touch()` function forces all the soft labels to be output the next time `slk_refresh()` or `slk_noutrefresh()` is called.

The `slk_attron()`, `slk_attrset()`, and `slk_attroff()` functions correspond to the `attron(3XCURSES)`, `attrset(3XCURSES)`, and `attroff(3XCURSES)` functions. They have an effect only if soft labels are stimulated on the bottom line of the screen.

The `slk_attr_on()`, `slk_attr_off()`, `slk_attr_set()` and `slk_color()` functions correspond to the `attr_on(3XCURSES)`, `attr_off(3XCURSES)`, `attr_set(3XCURSES)`, and `color_set(3XCURSES)` functions. As a result, they support color and the attribute constants with the `WA_` prefix.

slk_set(3XCURSES)

The *opts* argument is reserved for definition in a future release. Currently, the *opts* argument is a null pointer.

RETURN VALUES Upon successful completion, the `slk_label()` function returns the requested label with leading and trailing blanks stripped. Otherwise, it returns a null pointer.

Upon successful completion, the other functions return OK. Otherwise, they return ERR.

ERRORS No errors are defined.

USAGE When using multi-byte character sets, applications should check the width of the string by calling `mbstowcs(3C)` and then `wcswidth(3C)` before calling `slk_set()`. When using wide characters, applications should check the width of the string by calling `wcswidth()` before calling `slk_set()`.

Since the number of columns that a wide string will occupy is codeset-specific, call `wcwidth(3C)` and `wcswidth(3C)` to check the number of column positions in the string before calling `slk_wset()`.

Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attr_get(3XCURSES)`, `attroff(3XCURSES)`, `delscreen(3XCURSES)`, `libcurses(3XCURSES)`, `mbstowcs(3C)`, `ripoffline(3XCURSES)`, `wcswidth(3C)`, `wcwidth(3C)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | curs_slk, slk_init, slk_set, slk_refresh, slk_noutrefresh, slk_label, slk_clear, slk_restore, slk_touch, slk_attron, slk_attrset, slk_attroff – curses soft label routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int slk_init(int fmt); int slk_set(int labnum, char *label, int fmt); int slk_refresh(void); int slk_noutrefresh(void); char *slk_label(int labnum); int slk_clear(void); int slk_restore(void); int slk_touch(void); int slk_attron(chtype attrs); int slk_attrset(chtype attrs); int slk_attroff(chtype attrs);</pre> |
| DESCRIPTION | <p>curses manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, <i>curses</i> takes over the bottom line of <code>stdscr</code>, reducing the size of <code>stdscr</code> and the variable <code>LINES</code>. <i>curses</i> standardizes on eight labels of up to eight characters each.</p> <p>To use soft labels, the <code>slk_init()</code> routine must be called before <code>initscr()</code> or <code>newterm()</code> is called. If <code>initscr()</code> eventually uses a line from <code>stdscr</code> to emulate the soft labels, then <code>fmt</code> determines how the labels are arranged on the screen. Setting <code>fmt</code> to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement.</p> <p>With the <code>slk_set()</code> routine, <i>labnum</i> is the label number, from 1 to 8. <i>label</i> is the string to be put on the label, up to eight characters in length. A null string or a null pointer sets up a blank label. <code>fmt</code> is either 0, 1, or 2, indicating whether the label is to be left-justified, centered, or right-justified, respectively, within the label.</p> <p>The <code>slk_refresh()</code> and <code>slk_noutrefresh()</code> routines correspond to the <code>wrefresh()</code> and <code>wnoutrefresh()</code> routines.</p> <p>With the <code>slk_label()</code> routine, the current label for label number <i>labnum</i> is returned with leading and trailing blanks stripped.</p> <p>With the <code>slk_clear()</code> routine, the soft labels are cleared from the screen.</p> <p>With the <code>slk_restore()</code> routine, the soft labels are restored to the screen after a <code>slk_clear()</code> is performed.</p> |

slk_touch(3CURSES)

With the `slk_touch()` routine, all the soft labels are forced to be output the next time a `slk_noutrefresh()` is performed.

The `slk_attron()`, `slk_attrset()`, and `slk_attroff()` routines correspond to `attron()`, `attrset()`, and `attroff()`. They have an effect only if soft labels are simulated on the bottom line of the screen.

RETURN VALUES Routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`slk_label()` returns `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_attr(3CURSES)`, `curl_initscr(3CURSES)`, `curl_refresh(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.

| | | | | | | | | | | | | | | | |
|--------------------------|--|--------------|---|-------------|---|--------------------------|------------------|------------|---|---------------|----------------------------------|--------------|--|----------------|---|
| NAME | slk_attroff, slk_attr_off, slk_attron, slk_attr_on, slk_attrset, slk_attr_set, slk_clear, slk_color, slk_init, slk_label, slk_noutrefresh, slk_refresh, slk_restore, slk_set, slk_touch, slk_wset – soft label functions | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int slk_attroff(const chtype attrs); int slk_attr_off(const attr_t attrs, void *opts); int slk_attron(const chtype attrs); int slk_attr_on(const attr_t attrs, void *opts); int slk_attrset(const chtype attrs); int slk_attr_set(const attr_t attrs, short color_pair_number, void *opts); int slk_clear(void); int slk_color(short color_pair_number); int slk_init(int fmt); char *slk_label(int labnum); int slk_noutrefresh(void); int slk_refresh(void); int slk_restore(void); int slk_set(int labnum, const char *label, int justify); int slk_touch(void); int slk_wset(int labnum, const wchar_t *label, int justify);</pre> | | | | | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>attrs</i></td> <td>are the window attributes to be added or removed.</td> </tr> <tr> <td><i>opts</i></td> <td>Is reserved for future use. Currently, this must be a null pointer.</td> </tr> <tr> <td><i>color_pair_number</i></td> <td>Is a color pair.</td> </tr> <tr> <td><i>fmt</i></td> <td>Is the format of how the labels are arranged on the screen.</td> </tr> <tr> <td><i>labnum</i></td> <td>Is the number of the soft label.</td> </tr> <tr> <td><i>label</i></td> <td>Is the name to be given to a soft label.</td> </tr> <tr> <td><i>justify</i></td> <td>Is a number indicating how to justify the label name.</td> </tr> </table> | <i>attrs</i> | are the window attributes to be added or removed. | <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | <i>color_pair_number</i> | Is a color pair. | <i>fmt</i> | Is the format of how the labels are arranged on the screen. | <i>labnum</i> | Is the number of the soft label. | <i>label</i> | Is the name to be given to a soft label. | <i>justify</i> | Is a number indicating how to justify the label name. |
| <i>attrs</i> | are the window attributes to be added or removed. | | | | | | | | | | | | | | |
| <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | | | | | | | | | | | | | | |
| <i>color_pair_number</i> | Is a color pair. | | | | | | | | | | | | | | |
| <i>fmt</i> | Is the format of how the labels are arranged on the screen. | | | | | | | | | | | | | | |
| <i>labnum</i> | Is the number of the soft label. | | | | | | | | | | | | | | |
| <i>label</i> | Is the name to be given to a soft label. | | | | | | | | | | | | | | |
| <i>justify</i> | Is a number indicating how to justify the label name. | | | | | | | | | | | | | | |

slk_touch(3XCURSES)

DESCRIPTION

The Curses interface manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, Curses takes over the bottom line of *stdscr*, reducing the size of *stdscr* and the value of the `LINES` external variable. There can be up to eight labels of up to eight display columns each.

To use soft labels, `slk_init()` must be called before calling `initscr(3XCURSES)`, `newterm(3XCURSES)`, or `ripoffline(3XCURSES)`. If `initscr()` eventually uses a line from *stdscr* to emulate the soft labels, then *fmt* determines how the labels are arranged on the screen. Setting *fmt* to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement. Other values for *fmt* are unspecified.

The `slk_init()` function has the effect of calling `ripoffline()` to reserve one screen line to accommodate the requested format.

The `slk_set()` and `slk_wset()` functions specify the text of soft label number *labnum*, within the range from 1 to and including 8. The *label* argument is the string to be put the lable. With `slk_set()` and `slk_wset()`, the width of the label is limited to eight columns positions. A null string or a null pointer specifies a blank label. The *justify* argument can have the following values to indicate how to justify *label* within the space reserved for it:

- 0 Align the start of *label* with the start of the space
- 1 Center *label* within the space
- 2 Align the end of *label* with the end of the space

The `slk_refresh()` and `slk_noutrefresh()` functions correspond to the `wrefresh(3XCURSES)` and `wnoutrefresh(3XCURSES)` functions.

The `slk_label()` function obtains soft label number *labnum*.

The `slk_clear()` function immediately clears the soft labels from the screen.

The `slk_restore()` function immediately restores the soft labels to the screen after a call to `slk_clear()`.

The `slk_touch()` function forces all the soft labels to be output the next time `slk_refresh()` or `slk_noutrefresh()` is called.

The `slk_attron()`, `slk_attrset()`, and `slk_attroff()` functions correspond to the `attron(3XCURSES)`, `attrset(3XCURSES)`, and `attroff(3XCURSES)` functions. They have an effect only if soft labels are stimulated on the bottom line of the screen.

The `slk_attr_on()`, `slk_attr_off()`, `slk_attr_set()` and `slk_color()` functions correspond to the `attr_on(3XCURSES)`, `attr_off(3XCURSES)`, `attr_set(3XCURSES)`, and `color_set(3XCURSES)` functions. As a result, they support color and the attribute constants with the `WA_` prefix.

slk_touch(3XCURSES)

The *opts* argument is reserved for definition in a future release. Currently, the *opts* argument is a null pointer.

RETURN VALUES Upon successful completion, the `slk_label()` function returns the requested label with leading and trailing blanks stripped. Otherwise, it returns a null pointer.

Upon successful completion, the other functions return OK. Otherwise, they return ERR.

ERRORS No errors are defined.

USAGE When using multi-byte character sets, applications should check the width of the string by calling `mbstowcs(3C)` and then `wcswidth(3C)` before calling `slk_set()`. When using wide characters, applications should check the width of the string by calling `wcswidth()` before calling `slk_set()`.

Since the number of columns that a wide string will occupy is codeset-specific, call `wcwidth(3C)` and `wcswidth(3C)` to check the number of column positions in the string before calling `slk_wset()`.

Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attr_get(3XCURSES)`, `attroff(3XCURSES)`, `delscreen(3XCURSES)`, `libcurses(3XCURSES)`, `mbstowcs(3C)`, `ripcoffline(3XCURSES)`, `wcswidth(3C)`, `wcwidth(3C)`, `attributes(5)`, `standards(5)`

slk_wset(3XCURSES)

| | | | | | | | | | | | | | | | |
|--------------------------|--|--------------|---|-------------|---|--------------------------|------------------|------------|---|---------------|----------------------------------|--------------|--|----------------|---|
| NAME | slk_attroff, slk_attr_off, slk_attron, slk_attr_on, slk_attrset, slk_attr_set, slk_clear, slk_color, slk_init, slk_label, slk_noutrefresh, slk_refresh, slk_restore, slk_set, slk_touch, slk_wset – soft label functions | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int slk_attroff(const chtype attrs); int slk_attr_off(const attr_t attrs, void *opts); int slk_attron(const chtype attrs); int slk_attr_on(const attr_t attrs, void *opts); int slk_attrset(const chtype attrs); int slk_attr_set(const attr_t attrs, short color_pair_number, void *opts); int slk_clear(void); int slk_color(short color_pair_number); int slk_init(int fmt); char *slk_label(int labnum); int slk_noutrefresh(void); int slk_refresh(void); int slk_restore(void); int slk_set(int labnum, const char *label, int justify); int slk_touch(void); int slk_wset(int labnum, const wchar_t *label, int justify);</pre> | | | | | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>attrs</i></td><td>are the window attributes to be added or removed.</td></tr><tr><td><i>opts</i></td><td>Is reserved for future use. Currently, this must be a null pointer.</td></tr><tr><td><i>color_pair_number</i></td><td>Is a color pair.</td></tr><tr><td><i>fmt</i></td><td>Is the format of how the labels are arranged on the screen.</td></tr><tr><td><i>labnum</i></td><td>Is the number of the soft label.</td></tr><tr><td><i>label</i></td><td>Is the name to be given to a soft label.</td></tr><tr><td><i>justify</i></td><td>Is a number indicating how to justify the label name.</td></tr></table> | <i>attrs</i> | are the window attributes to be added or removed. | <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | <i>color_pair_number</i> | Is a color pair. | <i>fmt</i> | Is the format of how the labels are arranged on the screen. | <i>labnum</i> | Is the number of the soft label. | <i>label</i> | Is the name to be given to a soft label. | <i>justify</i> | Is a number indicating how to justify the label name. |
| <i>attrs</i> | are the window attributes to be added or removed. | | | | | | | | | | | | | | |
| <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | | | | | | | | | | | | | | |
| <i>color_pair_number</i> | Is a color pair. | | | | | | | | | | | | | | |
| <i>fmt</i> | Is the format of how the labels are arranged on the screen. | | | | | | | | | | | | | | |
| <i>labnum</i> | Is the number of the soft label. | | | | | | | | | | | | | | |
| <i>label</i> | Is the name to be given to a soft label. | | | | | | | | | | | | | | |
| <i>justify</i> | Is a number indicating how to justify the label name. | | | | | | | | | | | | | | |

DESCRIPTION | The Curses interface manipulates the set of soft function-key labels that exist on many terminals. For those terminals that do not have soft labels, Curses takes over the bottom line of *stdscr*, reducing the size of *stdscr* and the value of the `LINES` external variable. There can be up to eight labels of up to eight display columns each.

To use soft labels, `slk_init()` must be called before calling `initscr(3XCURSES)`, `newterm(3XCURSES)`, or `ripoffline(3XCURSES)`. If `initscr()` eventually uses a line from *stdscr* to emulate the soft labels, then *fmt* determines how the labels are arranged on the screen. Setting *fmt* to 0 indicates a 3-2-3 arrangement of the labels; 1 indicates a 4-4 arrangement. Other values for *fmt* are unspecified.

The `slk_init()` function has the effect of calling `ripoffline()` to reserve one screen line to accommodate the requested format.

The `slk_set()` and `slk_wset()` functions specify the text of soft label number *labnum*, within the range from 1 to and including 8. The *label* argument is the string to be put the lable. With `slk_set()` and `slk_wset()`, the width of the label is limited to eight columns positions. A null string or a null pointer specifies a blank label. The *justify* argument can have the following values to indicate how to justify *label* within the space reserved for it:

- 0 Align the start of *label* with the start of the space
- 1 Center *label* within the space
- 2 Align the end of *label* with the end of the space

The `slk_refresh()` and `slk_noutrefresh()` functions correspond to the `wrefresh(3XCURSES)` and `wnoutrefresh(3XCURSES)` functions.

The `slk_label()` function obtains soft label number *labnum*.

The `slk_clear()` function immediately clears the soft labels from the screen.

The `slk_restore()` function immediately restores the soft labels to the screen after a call to `slk_clear()`.

The `slk_touch()` function forces all the soft labels to be output the next time `slk_refresh()` or `slk_noutrefresh()` is called.

The `slk_attron()`, `slk_attrset()`, and `slk_attroff()` functions correspond to the `attron(3XCURSES)`, `attrset(3XCURSES)`, and `attroff(3XCURSES)` functions. They have an effect only if soft labels are stimulated on the bottom line of the screen.

The `slk_attr_on()`, `slk_attr_off()`, `slk_attr_set()` and `slk_color()` functions correspond to the `attr_on(3XCURSES)`, `attr_off(3XCURSES)`, `attr_set(3XCURSES)`, and `color_set(3XCURSES)` functions. As a result, they support color and the attribute constants with the `WA_` prefix.

slk_wset(3XCURSES)

The *opts* argument is reserved for definition in a future release. Currently, the *opts* argument is a null pointer.

RETURN VALUES Upon successful completion, the `slk_label()` function returns the requested label with leading and trailing blanks stripped. Otherwise, it returns a null pointer.

Upon successful completion, the other functions return OK. Otherwise, they return ERR.

ERRORS No errors are defined.

USAGE When using multi-byte character sets, applications should check the width of the string by calling `mbstowcs(3C)` and then `wcswidth(3C)` before calling `slk_set()`. When using wide characters, applications should check the width of the string by calling `wcswidth()` before calling `slk_set()`.

Since the number of columns that a wide string will occupy is codeset-specific, call `wcwidth(3C)` and `wcswidth(3C)` to check the number of column positions in the string before calling `slk_wset()`.

Most applications would use `slk_noutrefresh()` because a `wrefresh()` is likely to follow soon.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attr_get(3XCURSES)`, `attroff(3XCURSES)`, `delscreen(3XCURSES)`, `libcurses(3XCURSES)`, `mbstowcs(3C)`, `ripoffline(3XCURSES)`, `wcswidth(3C)`, `wcwidth(3C)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | plot, arc, box, circle, closepl, closevt, cont, erase, label, line, linemod, move, openpl, openvt, point, space – graphics interface |
| SYNOPSIS | <pre>cc [flag ...] file ... -lplot [library...] #include <plot.h> void arc(short x0, short y0, short x1, short y1, short x2, short y2); void box(short x0, short y0, short x1, short y1); void circle(short x, short y, short r); void closepl(); void closevt(); void cont(short x, short y); void erase(); void label(char *s); void line(short x0, short y0, short x1, short y1); void linemod(char *s); void move(short x, short y); void openpl(); void openvt(); void point(short x, short y); void space(short x0, short y0, short x1, short y1);</pre> |
| DESCRIPTION | <p>These functions generate graphics output for a set of output devices. The format of the output is dependent upon which link editor option is used when the program is compiled and linked (see Link Editor).</p> <p>The term "current point" refers to the current setting for the x and y coordinates.</p> <p>The <code>arc()</code> function specifies a circular arc. The coordinates $(x0, y0)$ specify the center of the arc. The coordinates $(x1, y1)$ specify the starting point of the arc. The coordinates $(x2, y2)$ specify the end point of the circular arc.</p> <p>The <code>box()</code> function specifies a rectangle with coordinates $(x0, y0)$, $(x0, y1)$, $(x1, y0)$, and $(x1, y1)$. The current point is set to $(x1, y1)$.</p> <p>The <code>circle()</code> function specifies a circle with a center at the coordinates (x, y) and a radius of r.</p> <p>The <code>closevt()</code> and <code>closepl()</code> functions flush the output.</p> <p>The <code>cont()</code> function specifies a line beginning at the current point and ending at the coordinates (x, y). The current point is set to (x, y).</p> |

space(3PLOT)

The `erase()` function starts another frame of output.

The `label()` function places the null terminated string `s` so that the first character falls on the current point. The string is then terminated by a NEWLINE character.

The `line()` function draws a line starting at the coordinates (x_0, y_0) and ending at the coordinates (x_1, y_1) . The current point is set to (x_1, y_1) .

The `linemod()` function specifies the style for drawing future lines. `s` may contain one of the following: `dotted`, `solid`, `longdashed`, `shortdashed`, or `dotdashed`.

The `move()` function sets the current point to the coordinates (x, y) .

The `openpl()` or `openvt()` function must be called to open the device before any other plot functions are called.

The `point()` function plots the point given by the coordinates (x, y) . The current point is set to (x, y) .

The `space()` function specifies the size of the plotting area. The plot will be reduced or enlarged as necessary to fit the area specified. The coordinates (x_0, y_0) specify the lower left hand corner of the plotting area. The coordinates (x_1, y_1) specify the upper right hand corner of the plotting area.

Link Editor Various flavors of these functions exist for different output devices. They are obtained by using the following `ld(1)` options:

| | |
|---------------------|--|
| <code>-lplot</code> | device-independent graphics stream on standard output in the format described in <code>plot(4B)</code> |
| <code>-l300</code> | GSI 300 terminal |
| <code>-l300s</code> | GSI 300S terminal |
| <code>-l4014</code> | Tektronix 4014 terminal |
| <code>-l450</code> | GSI 450 terminal |
| <code>-lvt0</code> | |

FILES

| | |
|--|----------------------|
| <code>/usr/lib/libplot.a</code> | archive library |
| <code>/usr/lib/libplot.so.1</code> | shared object |
| <code>/usr/lib/sparcv9/libplot.so.1</code> | 64-bit shared object |
| <code>/usr/lib/lib300.a</code> | archive library |
| <code>/usr/lib/lib300.so.1</code> | shared object |

```

/usr/lib/sparcv9/lib300.so.1
  64-bit shared object
/usr/lib/lib300s.a
  archive library
/usr/lib/lib300s.so.1
  shared object
/usr/lib/sparcv9/lib300s.so.1
  64-bit shared object
/usr/lib/lib4014.a
  archive library
/usr/lib/lib4014.so.1
  shared object
/usr/lib/sparcv9/lib4014.so.1
  64-bit shared object
/usr/lib/lib450.a
  archive library
/usr/lib/lib450.so.1
  shared object
/usr/lib/sparcv9/lib450.so.1
  64-bit shared object
/usr/lib/libvt0.a
  archive library
/usr/lib/libvt0.so.1
  shared object
/usr/lib/sparcv9/libvt0.so.1
  64-bit shared object

```

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `graph(1)`, `ld(1)`, `libplot(3LIB)`, `plot(4B)`, `attributes(5)`

standend(3CURSES)

| | | | | | | | | | | | | | | | |
|---------------------------|--|-------------------------|--|--------------------------|-------------|------------------------|---------------|----------------------|----------|--------------------|-------------|---------------------|----------------------|---------------------------|-------------------------|
| NAME | <code>curs_attr</code> , <code>attroff</code> , <code>wattroff</code> , <code>attron</code> , <code>wattron</code> , <code>attrset</code> , <code>wattrset</code> , <code>standend</code> , <code>wstandend</code> , <code>standout</code> , <code>wstandout</code> – curses character and window attribute control routines | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int attroff(int attrs); int wattroff(WINDOW *win, int attrs); int attron(int attrs); int wattron(WINDOW *win, int attrs); int attrset(int attrs); int wattrset(WINDOW *win, int attrs); int standend(void); int wstandend(WINDOW *win); int standout(void); int wstandout(WINDOW *win);</pre> | | | | | | | | | | | | | | |
| DESCRIPTION | <p>All of these routines manipulate the current attributes of the named window. The current attributes of a window are applied to all characters that are written into the window with <code>waddch()</code>, <code>waddstr()</code>, and <code>wprintw()</code>. Attributes are a property of the character, and move with the character through any scrolling and insert/delete line/character operations. To the extent possible on the particular terminal, they are displayed as the graphic rendition of characters put on the screen.</p> <p>The routine <code>attrset()</code> sets the current attributes of the given window to <i>attrs</i>. The routine <code>attroff()</code> turns off the named attributes without turning any other attributes on or off. The routine <code>attron()</code> turns on the named attributes without affecting any others. The routine <code>standout()</code> is the same as <code>attron(A_STANDOUT)</code>. The routine <code>standend()</code> is the same as <code>attroff()</code>, that is, it turns off all attributes.</p> | | | | | | | | | | | | | | |
| Attributes | <p>The following video attributes, defined in <code><curses.h></code>, can be passed to the routines <code>attron()</code>, <code>attroff()</code>, and <code>attrset()</code>, or OR-ed with the characters passed to <code>addch()</code>.</p> <table><tr><td><code>A_STANDOUT</code></td><td>Best highlighting mode of the terminal</td></tr><tr><td><code>A_UNDERLINE</code></td><td>Underlining</td></tr><tr><td><code>A_REVERSE</code></td><td>Reverse video</td></tr><tr><td><code>A_BLINK</code></td><td>Blinking</td></tr><tr><td><code>A_DIM</code></td><td>Half bright</td></tr><tr><td><code>A_BOLD</code></td><td>Extra bright or bold</td></tr><tr><td><code>A_ALTCHARSET</code></td><td>Alternate character set</td></tr></table> | <code>A_STANDOUT</code> | Best highlighting mode of the terminal | <code>A_UNDERLINE</code> | Underlining | <code>A_REVERSE</code> | Reverse video | <code>A_BLINK</code> | Blinking | <code>A_DIM</code> | Half bright | <code>A_BOLD</code> | Extra bright or bold | <code>A_ALTCHARSET</code> | Alternate character set |
| <code>A_STANDOUT</code> | Best highlighting mode of the terminal | | | | | | | | | | | | | | |
| <code>A_UNDERLINE</code> | Underlining | | | | | | | | | | | | | | |
| <code>A_REVERSE</code> | Reverse video | | | | | | | | | | | | | | |
| <code>A_BLINK</code> | Blinking | | | | | | | | | | | | | | |
| <code>A_DIM</code> | Half bright | | | | | | | | | | | | | | |
| <code>A_BOLD</code> | Extra bright or bold | | | | | | | | | | | | | | |
| <code>A_ALTCHARSET</code> | Alternate character set | | | | | | | | | | | | | | |

standend(3CURSES)

A_CHARTEXT Bit-mask to extract a character

COLOR_PAIR(*n*) Color-pair number *n*

The following macro is the reverse of COLOR_PAIR(*n*) :

PAIR_NUMBER(*attrs*) Returns the pair number associated with the COLOR_PAIR(*n*) attribute

RETURN VALUES These routines always return 1.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_addch(3CURSES), curs_addstr(3CURSES), curs_printw(3CURSES), curses(3CURSES), attributes(5)

NOTES The header < curses.h > automatically includes the headers < stdio.h > and < unctrl.h >.

Note that attroff(), wattroff(), attron(), wattron(), wattrset(), standend(), and standout() may be macros.

standend(3XCURSES)

| NAME | standend, standout, wstandend, wstandout – set/clear window attributes | | | | | | |
|----------------------|--|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int standend(void); int standout(void); int wstandend(WINDOW *win); int wstandout(WINDOW *win);</pre> | | | | | | |
| PARAMETERS | <i>win</i> Is a pointer to the window in which attribute changes are to be made. | | | | | | |
| DESCRIPTION | <p>The standend() and wstandend() functions turn off all attributes associated with stdscr and <i>win</i> respectively.</p> <p>The standout() and wstandout() functions turn on the A_STANDOUT attribute of stdscr and <i>win</i> respectively.</p> | | | | | | |
| RETURN VALUES | These functions always return 1. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | attr_get(3XCURSES), attroff(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5) | | | | | | |

| | | | | | | | | | | | | | | | |
|--------------------|--|------------|--|-------------|-------------|-----------|---------------|---------|----------|-------|-------------|--------|----------------------|--------------|-------------------------|
| NAME | curs_attr, attroff, wattroff, attron, wattron, attrset, wattrset, standend, wstandend, standout, wstandout – curses character and window attribute control routines | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int attroff(int attrs); int wattroff(WINDOW *win, int attrs); int attron(int attrs); int wattron(WINDOW *win, int attrs); int attrset(int attrs); int wattrset(WINDOW *win, int attrs); int standend(void); int wstandend(WINDOW *win); int standout(void); int wstandout(WINDOW *win);</pre> | | | | | | | | | | | | | | |
| DESCRIPTION | <p>All of these routines manipulate the current attributes of the named window. The current attributes of a window are applied to all characters that are written into the window with <code>waddch()</code>, <code>waddstr()</code>, and <code>wprintw()</code>. Attributes are a property of the character, and move with the character through any scrolling and insert/delete line/character operations. To the extent possible on the particular terminal, they are displayed as the graphic rendition of characters put on the screen.</p> <p>The routine <code>attrset()</code> sets the current attributes of the given window to <code>attrs</code>. The routine <code>attroff()</code> turns off the named attributes without turning any other attributes on or off. The routine <code>attron()</code> turns on the named attributes without affecting any others. The routine <code>standout()</code> is the same as <code>attron(A_STANDOUT)</code>. The routine <code>standend()</code> is the same as <code>attrset()</code>, that is, it turns off all attributes.</p> | | | | | | | | | | | | | | |
| Attributes | <p>The following video attributes, defined in <code><curses.h></code>, can be passed to the routines <code>attron()</code>, <code>attroff()</code>, and <code>attrset()</code>, or OR-ed with the characters passed to <code>addch()</code>.</p> <table border="0"> <tr> <td style="padding-right: 20px;">A_STANDOUT</td> <td>Best highlighting mode of the terminal</td> </tr> <tr> <td>A_UNDERLINE</td> <td>Underlining</td> </tr> <tr> <td>A_REVERSE</td> <td>Reverse video</td> </tr> <tr> <td>A_BLINK</td> <td>Blinking</td> </tr> <tr> <td>A_DIM</td> <td>Half bright</td> </tr> <tr> <td>A_BOLD</td> <td>Extra bright or bold</td> </tr> <tr> <td>A_ALTCHARSET</td> <td>Alternate character set</td> </tr> </table> | A_STANDOUT | Best highlighting mode of the terminal | A_UNDERLINE | Underlining | A_REVERSE | Reverse video | A_BLINK | Blinking | A_DIM | Half bright | A_BOLD | Extra bright or bold | A_ALTCHARSET | Alternate character set |
| A_STANDOUT | Best highlighting mode of the terminal | | | | | | | | | | | | | | |
| A_UNDERLINE | Underlining | | | | | | | | | | | | | | |
| A_REVERSE | Reverse video | | | | | | | | | | | | | | |
| A_BLINK | Blinking | | | | | | | | | | | | | | |
| A_DIM | Half bright | | | | | | | | | | | | | | |
| A_BOLD | Extra bright or bold | | | | | | | | | | | | | | |
| A_ALTCHARSET | Alternate character set | | | | | | | | | | | | | | |

standout(3CURSES)

A_CHARTEXT Bit-mask to extract a character

COLOR_PAIR(*n*) Color-pair number *n*

The following macro is the reverse of COLOR_PAIR(*n*) :

PAIR_NUMBER(*attrs*) Returns the pair number associated with the
COLOR_PAIR(*n*) attribute

RETURN VALUES These routines always return 1.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_addch(3CURSES), curs_addstr(3CURSES), curs_printw(3CURSES),
curses(3CURSES), attributes(5)

NOTES The header < curses . h > automatically includes the headers < stdio . h > and
< unctrl . h >.

Note that attroff () , wattroff () , attron () , wattron () , wattrset () ,
standend () , and standout () may be macros.

| NAME | standend, standout, wstandend, wstandout – set/clear window attributes | | | | | | |
|----------------------|--|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int standend(void); int standout(void); int wstandend(WINDOW *win); int wstandout(WINDOW *win);</pre> | | | | | | |
| PARAMETERS | <p><i>win</i> Is a pointer to the window in which attribute changes are to be made.</p> | | | | | | |
| DESCRIPTION | <p>The standend() and wstandend() functions turn off all attributes associated with stdscr and <i>win</i> respectively.</p> <p>The standout() and wstandout() functions turn on the A_STANDOUT attribute of stdscr and <i>win</i> respectively.</p> | | | | | | |
| RETURN VALUES | These functions always return 1. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | attr_get(3XCURSES), attroff(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5) | | | | | | |

start_color(3CURSES)

| | |
|-----------------------------|---|
| NAME | <code> curs_color, start_color, init_pair, init_color, has_colors, can_change_color, color_content, pair_content</code> – curses color manipulation routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int start_color(void); int init_pair(short pair, short fg, short bg); int init_color(short color, short red, short green, short blue); bool has_colors(void); bool can_change_color(void); int color_content(short color, short *redp, short *greenp, short *bluep); int pair_content(short pair, short *fgp, short *bgp);</pre> |
| Overview | <p>curses provides routines that manipulate color on color alphanumeric terminals. To use these routines <code>start_color()</code> must be called, usually right after <code>initscr()</code>. See <code> curs_initscr(3CURSES)</code>. Colors are always used in pairs (referred to as color-pairs). A color-pair consists of a foreground color (for characters) and a background color (for the field on which the characters are displayed). A programmer initializes a color-pair with the routine <code>init_pair</code>. After it has been initialized, <code>COLOR_PAIR(n)</code>, a macro defined in <code><curses.h></code>, can be used in the same ways other video attributes can be used. If a terminal is capable of redefining colors, the programmer can use the routine <code>init_color()</code> to change the definition of a color. The routines <code>has_colors()</code> and <code>can_change_color()</code> return <code>TRUE</code> or <code>FALSE</code>, depending on whether the terminal has color capabilities and whether the programmer can change the colors. The routine <code>color_content()</code> allows a programmer to identify the amounts of red, green, and blue components in an initialized color. The routine <code>pair_content()</code> allows a programmer to find out how a given color-pair is currently defined.</p> |
| Routine Descriptions | <p>The <code>start_color()</code> routine requires no arguments. It must be called if the programmer wants to use colors, and before any other color manipulation routine is called. It is good practice to call this routine right after <code>initscr()</code>. <code>start_color()</code> initializes eight basic colors (black, red, green, yellow, blue, magenta, cyan, and white), and two global variables, <code>COLORS</code> and <code>COLOR_PAIRS</code> (respectively defining the maximum number of colors and color-pairs the terminal can support). It also restores the colors on the terminal to the values they had when the terminal was just turned on.</p> <p>The <code>init_pair()</code> routine changes the definition of a color-pair. It takes three arguments: the number of the color-pair to be changed, the foreground color number, and the background color number. The value of the first argument must be between 1 and <code>COLOR_PAIRS-1</code>. The value of the second and third arguments must be between 0 and <code>COLORS</code>. If the color-pair was previously initialized, the screen is refreshed and all occurrences of that color-pair is changed to the new definition.</p> |

start_color(3CURSES)

The `init_color()` routine changes the definition of a color. It takes four arguments: the number of the color to be changed followed by three RGB values (for the amounts of red, green, and blue components). The value of the first argument must be between 0 and `COLORS`. (See the section `COLORS` for the default color index.) Each of the last three arguments must be a value between 0 and 1000. When `init_color()` is used, all occurrences of that color on the screen immediately change to the new definition.

The `has_colors()` routine requires no arguments. It returns `TRUE` if the terminal can manipulate colors; otherwise, it returns `FALSE`. This routine facilitates writing terminal-independent programs. For example, a programmer can use it to decide whether to use color or some other video attribute.

The `can_change_color()` routine requires no arguments. It returns `TRUE` if the terminal supports colors and can change their definitions; other, it returns `FALSE`. This routine facilitates writing terminal-independent programs.

The `color_content()` routine gives users a way to find the intensity of the red, green, and blue (RGB) components in a color. It requires four arguments: the color number, and three addresses of `shorts` for storing the information about the amounts of red, green, and blue components in the given color. The value of the first argument must be between 0 and `COLORS`. The values that are stored at the addresses pointed to by the last three arguments are between 0 (no component) and 1000 (maximum amount of component).

The `pair_content()` routine allows users to find out what colors a given color-pair consists of. It requires three arguments: the color-pair number, and two addresses of `shorts` for storing the foreground and the background color numbers. The value of the first argument must be between 1 and `COLOR_PAIRS-1`. The values that are stored at the addresses pointed to by the second and third arguments are between 0 and `COLORS`.

Colors In `< curses.h >` the following macros are defined. These are the default colors. `curses` also assumes that `COLOR_BLACK` is the default background color for all terminals.

```
COLOR_BLACK
COLOR_RED
COLOR_GREEN
COLOR_YELLOW
COLOR_BLUE
COLOR_MAGENTA
COLOR_CYAN
COLOR_WHITE
```

RETURN VALUES All routines that return an integer return `ERR` upon failure and `OK` upon successful completion.

start_color(3CURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_attr(3CURSES)`, `curl_initscr(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

start_color(3XCURSES)

| | |
|------------------------------|---|
| NAME | can_change_color, color_content, COLOR_PAIR, has_colors, init_color, init_pair, pair_content, PAIR_NUMBER, start_color, COLOR_PAIRS, COLORS – manipulate color information |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> bool can_change_color(void); int color_content(short color, short *red, short *green, short *blue); int COLOR_PAIR(int n); bool has_colors(void); int init_color(short color, short red, short green, short blue); int init_pair(short pair, short f, short b); int pair_content(short pair, short *f, short *b); int PAIR_NUMBER(int value); int start_color(void); extern int COLOR_PAIRS; extern int COLORS;</pre> |
| DESCRIPTION | These functions manipulate color on terminals that support color. |
| Querying Capabilities | The has_colors() function indicates whether the terminal is a color terminal. The can_change_color() function indicates whether the terminal is a color terminal on which colors can be redefined. |
| Initialization | <p>The start_color() function must be called to enable use of colors and before any color manipulation function is called. The function initializes eight basic colors (black, red, green, yellow, blue, magenta, cyan, and white) that can be specified by the color macros (such as COLOR_BLACK) defined in <curses.h>. The initial appearance of these colors is unspecified.</p> <p>The function also initializes two global external variables:</p> <ul style="list-style-type: none">■ COLORS defines the number of colors that the terminal supports. See Color Identification below. If COLORS is 0, the terminal does not support redefinition of colors and can_change_color() will return FALSE.■ COLOR_PAIRS defines the maximum number of color-pairs that the terminal supports. See User-defined Color Pairs below. |

start_color(3XCURSES)

| | | | | | | | | | | | | | | | |
|---------------------------------|--|--------------|--|------------|---|--------------|---|-------------|--|----------|--------------------------------|-------------|--|----------|---|
| | <p>The <code>start_color()</code> function also restores the colors on the terminal to terminal-specific initial values. The initial background color is assumed to be black for all terminals.</p> | | | | | | | | | | | | | | |
| Color Identification | <p>The <code>init_color()</code> function redefines color number <i>color</i>, on terminals that support the redefinition of colors, to have the red, green, and blue intensity components specified by <i>red</i>, <i>green</i>, and <i>blue</i>, respectively. Calling <code>init_color()</code> also changes all occurrences of the specified color on the screen to the new definition.</p> <p>The <code>color_content()</code> function identifies the intensity components of color number <i>color</i>. It stores the red, green, and blue intensity components of this color in the addresses pointed to by <i>red</i>, <i>green</i>, and <i>blue</i>, respectively.</p> <p>For both functions, the <i>color</i> argument must be in the range from 0 to and including <code>COLORS-1</code>. Valid intensity value range from 0 (no intensity component) up to and including 1000 (maximum intensity in that component).</p> | | | | | | | | | | | | | | |
| User-defined Color Pairs | <p>Calling <code>init_pair()</code> defines or redefines color-pair number <i>pair</i> to have foreground color <i>f</i> and background color <i>b</i>. Calling <code>init_pair()</code> changes any characters that were displayed in the color pair's old definition to the new definition and refreshes the screen.</p> <p>After defining the color pair, the macro <code>COLOR_PAIR(n)</code> returns the value of color pair <i>n</i>. This value is the color attribute as it would be extracted from a <code>chtype</code>. Controversy, the macro <code>COLOR_NUMBER(value)</code> returns the color pair number associated with the color attribute <i>value</i>.</p> <p>The <code>pair_content()</code> retrieves the component colors of a color-pair number <i>pair</i>. It stores the foreground and background color numbers in the variables pointed to by <i>f</i> and <i>b</i>, respectively.</p> <p>With <code>init_pair()</code> and <code>pair_content()</code>, the value of <i>pair</i> must be in a range from 0 to and including <code>COLOR_PAIRS-1</code>. Valid values for <i>f</i> and <i>b</i> are the range from 0 to and including <code>COLORS-1</code>.</p> | | | | | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>color</i></td><td>Is the number of the color for which to provide information (0 to <code>COLORS-1</code>).</td></tr><tr><td><i>red</i></td><td>Is a pointer to the RGB value for the amount of red in <i>color</i>.</td></tr><tr><td><i>green</i></td><td>Is a pointer to the RGB value for the amount of green in <i>color</i>.</td></tr><tr><td><i>blue</i></td><td>Is a pointer to the RGB value for the amount of blue in <i>color</i>.</td></tr><tr><td><i>n</i></td><td>Is the number of a color pair.</td></tr><tr><td><i>pair</i></td><td>Is the number of the color pair for which to provide information (1 to <code>COLOR_PAIRS-1</code>).</td></tr><tr><td><i>f</i></td><td>Is a pointer to the number of the foreground color (0 to <code>COLORS-1</code>) in <i>pair</i>.</td></tr></table> | <i>color</i> | Is the number of the color for which to provide information (0 to <code>COLORS-1</code>). | <i>red</i> | Is a pointer to the RGB value for the amount of red in <i>color</i> . | <i>green</i> | Is a pointer to the RGB value for the amount of green in <i>color</i> . | <i>blue</i> | Is a pointer to the RGB value for the amount of blue in <i>color</i> . | <i>n</i> | Is the number of a color pair. | <i>pair</i> | Is the number of the color pair for which to provide information (1 to <code>COLOR_PAIRS-1</code>). | <i>f</i> | Is a pointer to the number of the foreground color (0 to <code>COLORS-1</code>) in <i>pair</i> . |
| <i>color</i> | Is the number of the color for which to provide information (0 to <code>COLORS-1</code>). | | | | | | | | | | | | | | |
| <i>red</i> | Is a pointer to the RGB value for the amount of red in <i>color</i> . | | | | | | | | | | | | | | |
| <i>green</i> | Is a pointer to the RGB value for the amount of green in <i>color</i> . | | | | | | | | | | | | | | |
| <i>blue</i> | Is a pointer to the RGB value for the amount of blue in <i>color</i> . | | | | | | | | | | | | | | |
| <i>n</i> | Is the number of a color pair. | | | | | | | | | | | | | | |
| <i>pair</i> | Is the number of the color pair for which to provide information (1 to <code>COLOR_PAIRS-1</code>). | | | | | | | | | | | | | | |
| <i>f</i> | Is a pointer to the number of the foreground color (0 to <code>COLORS-1</code>) in <i>pair</i> . | | | | | | | | | | | | | | |

start_color(3XCURSES)

b Is a pointer to the number of the background color (0 to COLORS-1) in *pair*.

value Is a color attribute value.

RETURN VALUES The has_colors() function returns TRUE if the terminal can manipulate colors. Otherwise, it returns FALSE.

The can_change_color() function returns TRUE if the terminal supports colors and is able to change their definitions. Otherwise, it returns FALSE.

Upon successful completion, the other functions return OK. Otherwise, they return ERR.

ERRORS No errors are defined.

USAGE To use these functions, start_color() must be called, usually right after initscr(3XCURSES).

The can_change_color() and has_colors() functions facilitate writing terminal-independent applications. For example, a programmer can use them to decide whether to use color or some other video attribute.

On color terminals, a typical value of COLORS is 8 and the macros such as COLOR_BLACK return a value within the range from 0 to and including 7. However, applications cannot rely on this to be true.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO attroff(3XCURSES), delscreen(3XCURSES), initscr(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

stdscr(3XCURSES)

NAME | stdscr – default window

SYNOPSIS |

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \  
-R /usr/xpg4/lib -lcurses [ library... ]
```

|

```
c89 [ flag... ] file... -lcurses [ library... ]
```

|

```
#include <curses.h>
```

|

```
extern WINDOW *stdscr;
```

DESCRIPTION | The external variable `stdscr` specifies the default window used by functions that do not specify a window using an argument of type `WINDOW *`. Other windows may be created using `newwin()`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | `derwin(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | curs_pad, newpad, subpad, prefresh, pnoutrefresh, pechochar, pechowchar – create and display curses pads |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> WINDOW *newpad(int nlines, int ncols); WINDOW *subpad(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); int prefresh(WINDOW *pad, int pminrow, int pmincol, int sminrow, int smincol, int smaxrow, int smaxcol); int pnoutrefresh(WINDOW *pad, int pminrow, int pmincol, int sminrow, int smincol, int smaxrow, int smaxcol); int pechochar(WINDOW *pad, chtype ch); int pechowchar(WINDOW *pad, chtype wch);</pre> |
| DESCRIPTION | <p>The <code>newpad()</code> routine creates and returns a pointer to a new pad data structure with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. A pad is like a window, except that it is not restricted by the screen size, and is not necessarily associated with a particular part of the screen. Pads can be used when a large window is needed, and only a part of the window will be on the screen at one time. Automatic refreshes of pads (for example, from scrolling or echoing of input) do not occur. It is not legal to call <code>wrefresh(3CURSES)</code> with a <i>pad</i> as an argument; the routines <code>prefresh()</code> or <code>pnoutrefresh()</code> should be called instead. Note that these routines require additional parameters to specify the part of the pad to be displayed and the location on the screen to be used for the display.</p> <p>The <code>subpad()</code> routine creates and returns a pointer to a subwindow within a pad with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. Unlike <code>subwin(3CURSES)</code>, which uses screen coordinates, the window is at position (<i>begin_x</i>, <i>begin_y</i>) on the pad. The window is made in the middle of the window <i>orig</i>, so that changes made to one window affect both windows. During the use of this routine, it will often be necessary to call <code>touchwin(3CURSES)</code> or <code>touchline(3CURSES)</code> on <i>orig</i> before calling <code>prefresh()</code>.</p> <p>The <code>prefresh()</code> and <code>pnoutrefresh()</code> routines are analogous to <code>wrefresh(3CURSES)</code> and <code>wnoutrefresh(3CURSES)</code> except that they relate to pads instead of windows. The additional parameters are needed to indicate what part of the pad and screen are involved. <i>pminrow</i> and <i>pmincol</i> specify the upper left-hand corner of the rectangle to be displayed in the pad. <i>sminrow</i>, <i>smincol</i>, <i>smaxrow</i>, and <i>smaxcol</i> specify the edges of the rectangle to be displayed on the screen. The lower right-hand corner of the rectangle to be displayed in the pad is calculated from the screen coordinates, since the rectangles must be the same size. Both rectangles must be entirely contained within their respective structures. Negative values of <i>pminrow</i>, <i>pmincol</i>, <i>sminrow</i>, or <i>smincol</i> are treated as if they were zero.</p> |

subpad(3CURSES)

The `pechochar()` routine is functionally equivalent to a call to `addch(3CURSES)` followed by a call to `refresh(3CURSES)`, a call to `waddch(3CURSES)` followed by a call to `wrefresh(3CURSES)`, or a call to `waddch(3CURSES)` followed by a call to `prefresh()`. The knowledge that only a single character is being output is taken into consideration and, for non-control characters, a considerable performance gain might be seen by using these routines instead of their equivalents. In the case of `pechochar()`, the last location of the pad on the screen is reused for the arguments to `prefresh()`.

RETURN VALUES Routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

Routines that return pointers return `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `addch(3CURSES)`, `curses(3CURSES)`, `refresh(3CURSES)`, `subwin(3CURSES)`, `touchline(3CURSES)`, `touchwin(3CURSES)`, `waddch(3CURSES)`, `wnoutrefresh(3CURSES)`, `wrefresh(3CURSES)`, `attributes(5)`

NOTES The header file `<curses.h>` automatically includes the header files `<stdio.h>`, `<unctrl.h>` and `<widec.h>`.

Note that `pechochar()` may be a macro.

| | | | | | | | | | | | | | | | | | | | | | |
|--------------------|---|---------------|--|--------------|--|------------|-------------------------------------|----------------|--|----------------|--|----------------|---|----------------|--|----------------|--|----------------|---|-------------|---|
| NAME | newpad, pnoutrefresh, prefresh, subpad – create or refresh a pad or subpad | | | | | | | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> WINDOW *newpad(int nlines, int ncols); int pnoutrefresh(WINDOW *pad, int pminrow, int pmincol, int sminrow, int smincol, int smaxrow, int smaxcol); int prefresh(WINDOW *pad, int pminrow, int pmincol, int sminrow, int smincol, int smaxrow, int smaxcol); WINDOW *subpad(WINDOW *orig, int nlines, int ncols);</pre> | | | | | | | | | | | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="vertical-align: top;"><i>nlines</i></td> <td>Is the number of lines in the pad to be created.</td> </tr> <tr> <td style="vertical-align: top;"><i>ncols</i></td> <td>Is the number of columns in the pad to be created.</td> </tr> <tr> <td style="vertical-align: top;"><i>pad</i></td> <td>Is a pointer to the pad to refresh.</td> </tr> <tr> <td style="vertical-align: top;"><i>pminrow</i></td> <td>Is the row coordinate of the upper left corner of the pad rectangle to be copied</td> </tr> <tr> <td style="vertical-align: top;"><i>pmincol</i></td> <td>Is the column coordinate of the upper left corner of the pad rectangle to be copied.</td> </tr> <tr> <td style="vertical-align: top;"><i>sminrow</i></td> <td>Is the row coordinate of the upper left corner of the rectangle on the physical screen where pad is to be positioned.</td> </tr> <tr> <td style="vertical-align: top;"><i>smincol</i></td> <td>Is the column coordinate of the upper left corner of the rectangle on the physical screen where pad is to be positioned.</td> </tr> <tr> <td style="vertical-align: top;"><i>smaxrow</i></td> <td>Is the row coordinate of the lower right corner of the rectangle on the physical screen where the pad is to be positioned.</td> </tr> <tr> <td style="vertical-align: top;"><i>smaxcol</i></td> <td>Is the column coordinate of the lower right corner of the rectangle on the physical screen where the pad is to be positioned.</td> </tr> <tr> <td style="vertical-align: top;"><i>orig</i></td> <td>Is a pointer to the parent pad within which a sub-pad is created.</td> </tr> </table> | <i>nlines</i> | Is the number of lines in the pad to be created. | <i>ncols</i> | Is the number of columns in the pad to be created. | <i>pad</i> | Is a pointer to the pad to refresh. | <i>pminrow</i> | Is the row coordinate of the upper left corner of the pad rectangle to be copied | <i>pmincol</i> | Is the column coordinate of the upper left corner of the pad rectangle to be copied. | <i>sminrow</i> | Is the row coordinate of the upper left corner of the rectangle on the physical screen where pad is to be positioned. | <i>smincol</i> | Is the column coordinate of the upper left corner of the rectangle on the physical screen where pad is to be positioned. | <i>smaxrow</i> | Is the row coordinate of the lower right corner of the rectangle on the physical screen where the pad is to be positioned. | <i>smaxcol</i> | Is the column coordinate of the lower right corner of the rectangle on the physical screen where the pad is to be positioned. | <i>orig</i> | Is a pointer to the parent pad within which a sub-pad is created. |
| <i>nlines</i> | Is the number of lines in the pad to be created. | | | | | | | | | | | | | | | | | | | | |
| <i>ncols</i> | Is the number of columns in the pad to be created. | | | | | | | | | | | | | | | | | | | | |
| <i>pad</i> | Is a pointer to the pad to refresh. | | | | | | | | | | | | | | | | | | | | |
| <i>pminrow</i> | Is the row coordinate of the upper left corner of the pad rectangle to be copied | | | | | | | | | | | | | | | | | | | | |
| <i>pmincol</i> | Is the column coordinate of the upper left corner of the pad rectangle to be copied. | | | | | | | | | | | | | | | | | | | | |
| <i>sminrow</i> | Is the row coordinate of the upper left corner of the rectangle on the physical screen where pad is to be positioned. | | | | | | | | | | | | | | | | | | | | |
| <i>smincol</i> | Is the column coordinate of the upper left corner of the rectangle on the physical screen where pad is to be positioned. | | | | | | | | | | | | | | | | | | | | |
| <i>smaxrow</i> | Is the row coordinate of the lower right corner of the rectangle on the physical screen where the pad is to be positioned. | | | | | | | | | | | | | | | | | | | | |
| <i>smaxcol</i> | Is the column coordinate of the lower right corner of the rectangle on the physical screen where the pad is to be positioned. | | | | | | | | | | | | | | | | | | | | |
| <i>orig</i> | Is a pointer to the parent pad within which a sub-pad is created. | | | | | | | | | | | | | | | | | | | | |
| DESCRIPTION | The newpad() function creates a new pad with the specified number of lines and columns. A pointer to the new pad structure is returned. A pad differs from a window in that it is not restricted to the size of the physical screen. It is useful when only part of a large window will be displayed at any one time. | | | | | | | | | | | | | | | | | | | | |

subpad(3XCURSES)

Automatic refreshes by scrolling or echoing of input do not take place when pads are used. Pads have their own refresh commands, `prefresh()` and `pnoutrefresh()`.

The `prefresh()` function copies the specified portion of the logical pad to the terminal screen. The parameters `pmincol` and `pminrow` specify the upper left corner of the rectangular area of the pad to be displayed. The lower right coordinate of the rectangular area of the pad that is to be displayed is calculated from the screen parameters (`sminrow`, `smincol`, `smaxrow`, `smaxcol`).

This function calls the `pnoutrefresh()` function to copy the specified portion of *pad* to the terminal screen and the `doupdate(3XCURSES)` function to do the actual update. The logical cursor is copied to the same location in the physical window unless `leaveok(3XCURSES)` is enabled (in which case, the cursor is placed in a position that the program finds convenient).

When outputting several pads at once, it is often more efficient to call the `pnoutrefresh()` and `doupdate()` functions directly. A call to `pnoutrefresh()` for each pad first, followed by only one call to `doupdate()` to update the screen, results in one burst of output, fewer characters sent, and less CPU time used.

The `subpad()` function creates a sub-pad within the pad *orig* with the specified number of lines and columns. A pointer to the new pad structure is returned. The sub-pad is positioned in the middle of *orig*. Any changes made to one pad affect the other. `touchwin(3XCURSES)` or `touchline(3XCURSES)` will likely have to be called on pad *orig* to correctly update the window.

RETURN VALUES

On success, the `newpad()` and `subpad()` functions returns a pointer to the new pad data structure. Otherwise, they return a null pointer.

On success, the `pnoutrefresh()` and `prefresh()` functions return `OK`. Otherwise, they return `ERR`.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO

`clearok(3XCURSES)`, `doupdate(3XCURSES)`, `is_linetouched(3XCURSES)`, `libcurses(3XCURSES)`, `pechochar(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | curs_window, newwin, delwin, mvwin, subwin, derwin, mvderwin, dupwin, wsyncup, syncok, wcursyncup, wsyncdown – create curses windows |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> WINDOW *newwin(int nlines, int ncols, int begin_y, int begin_x); int delwin(WINDOW *win); int mvwin(WINDOW *win, int y, int x); WINDOW *subwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); WINDOW *derwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); int mvderwin(WINDOW *win, int par_y, int par_x); WINDOW *dupwin(WINDOW *win); void wsyncup(WINDOW *win); int syncok(WINDOW *win, bool bf); void wcursyncup(WINDOW *win); void wsyncdown(WINDOW *win);</pre> |
| DESCRIPTION | <p>The <code>newwin()</code> routine creates and returns a pointer to a new window with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. The upper left-hand corner of the window is at line <i>begin_y</i>, column <i>begin_x</i>. If either <i>nlines</i> or <i>ncols</i> is zero, they default to LINES — <i>begin_y</i> and COLS — <i>begin_x</i>. A new full-screen window is created by calling <code>newwin(0, 0, 0, 0)</code>.</p> <p>The <code>delwin()</code> routine deletes the named window, freeing all memory associated with it. Subwindows must be deleted before the main window can be deleted.</p> <p>The <code>mvwin()</code> routine moves the window so that the upper left-hand corner is at position (<i>x</i>, <i>y</i>). If the move would cause the window to be off the screen, it is an error and the window is not moved. Moving subwindows is allowed, but should be avoided.</p> <p>The <code>subwin()</code> routine creates and returns a pointer to a new window with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. The window is at position (<i>begin_y</i>, <i>begin_x</i>) on the screen. (This position is relative to the screen, and not to the window <i>orig</i>.) The window is made in the middle of the window <i>orig</i>, so that changes made to one window will affect both windows. The subwindow shares memory with the window <i>orig</i>. When using this routine, it is necessary to call <code>touchwin()</code> or <code>touchline()</code> on <i>orig</i> before calling <code>wrefresh()</code> on the subwindow.</p> <p>The <code>derwin()</code> routine is the same as <code>subwin()</code>, except that <i>begin_y</i> and <i>begin_x</i> are relative to the origin of the window <i>orig</i> rather than the screen. There is no difference between the subwindows and the derived windows.</p> |

subwin(3CURSES)

The `mvderwin()` routine moves a derived window (or subwindow) inside its parent window. The screen-relative parameters of the window are not changed. This routine is used to display different parts of the parent window at the same physical position on the screen.

The `dupwin()` routine creates an exact duplicate of the window *win*.

Each `curses` window maintains two data structures: the character image structure and the status structure. The character image structure is shared among all windows in the window hierarchy (that is, the window with all subwindows). The status structure, which contains information about individual line changes in the window, is private to each window. The routine `wrefresh()` uses the status data structure when performing screen updating. Since status structures are not shared, changes made to one window in the hierarchy may not be properly reflected on the screen.

The routine `wsyncup()` causes the changes in the status structure of a window to be reflected in the status structures of its ancestors. If `syncok()` is called with second argument `TRUE` then `wsyncup()` is called automatically whenever there is a change in the window.

The routine `wcursyncup()` updates the current cursor position of all the ancestors of the window to reflect the current cursor position of the window.

The routine `wsyncdown()` updates the status structure of the window to reflect the changes in the status structures of its ancestors. Applications seldom call this routine because it is called automatically by `wrefresh()`.

RETURN VALUES

Routines that return an integer return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`delwin()` returns the integer `ERR` upon failure and `OK` upon successful completion.

Routines that return pointers return `NULL` on error.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_refresh(3CURSES)`, `curs_touch(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

If many small changes are made to the window, the `wsyncup()` option could degrade performance.

`subwin(3CURSES)`

Note that `syncok()` may be a macro.

subwin(3XCURSES)

| | | | | | | | | | |
|--------------------|--|-------------|--|---------------|--|--------------|--|----------------|---|
| NAME | derwin, newwin, subwin – create a new window or subwindow | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> WINDOW *derwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); WINDOW *newwin(int nlines, int ncols, int begin_y, int begin_x); WINDOW *subwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x);</pre> | | | | | | | | |
| DESCRIPTION | <p>The <code>derwin()</code> function creates a subwindow within window <i>orig</i>, with the specified number of lines and columns, and upper left corner positioned at <i>begin_x</i>, <i>begin_y</i> relative to window <i>orig</i>. A pointer to the new window structure is returned.</p> <p>The <code>newwin()</code> function creates a new window with the specified number of lines and columns and upper left corner positioned at <i>begin_x</i>, <i>begin_y</i>. A pointer to the new window structure is returned. A full-screen window can be created by calling <code>newwin(0, 0, 0, 0)</code>.</p> <p>If the number of lines specified is zero, <code>newwin()</code> uses a default value of <code>LINES</code> minus <i>begin_y</i>; if the number of columns specified is zero, <code>newwin()</code> uses the default value of <code>COLS</code> minus <i>begin_x</i>.</p> <p>The <code>subwin()</code> function creates a subwindow within window <i>orig</i>, with the specified number of lines and columns, and upper left corner positioned at <i>begin_x</i>, <i>begin_y</i> (relative to the physical screen, <i>not</i> to window <i>orig</i>). A pointer to the new window structure is returned.</p> <p>The original window and subwindow share character storage of the overlapping area (each window maintains its own pointers, cursor location, and other items). This means that characters and attributes are identical in overlapping areas regardless of which window characters are written to.</p> <p>When using subwindows, it is often necessary to call <code>touchwin(3XCURSES)</code> before <code>wrefresh(3XCURSES)</code> to maintain proper screen contents.</p> | | | | | | | | |
| PARAMETERS | <table><tr><td><i>orig</i></td><td>Is a pointer to the parent window for the newly created subwindow.</td></tr><tr><td><i>nlines</i></td><td>Is the number of lines in the subwindow.</td></tr><tr><td><i>ncols</i></td><td>Is the number of columns in the subwindow.</td></tr><tr><td><i>begin_y</i></td><td>Is the y (row) coordinate of the upper left corner of the subwindow, relative to the parent window.</td></tr></table> | <i>orig</i> | Is a pointer to the parent window for the newly created subwindow. | <i>nlines</i> | Is the number of lines in the subwindow. | <i>ncols</i> | Is the number of columns in the subwindow. | <i>begin_y</i> | Is the y (row) coordinate of the upper left corner of the subwindow, relative to the parent window. |
| <i>orig</i> | Is a pointer to the parent window for the newly created subwindow. | | | | | | | | |
| <i>nlines</i> | Is the number of lines in the subwindow. | | | | | | | | |
| <i>ncols</i> | Is the number of columns in the subwindow. | | | | | | | | |
| <i>begin_y</i> | Is the y (row) coordinate of the upper left corner of the subwindow, relative to the parent window. | | | | | | | | |

subwin(3XCURSES)

begin_x Is the x (column) coordinate of the upper left corner of the subwindow, relative to the parent window.

RETURN VALUES On success, these functions return a pointer to the newly-created window. Otherwise, they return `ERR`.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `douupdate(3XCURSES)`, `is_linetouched(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

syncok(3CURSES)

| | |
|--------------------|--|
| NAME | <code>curs_window</code> , <code>newwin</code> , <code>delwin</code> , <code>mvwin</code> , <code>subwin</code> , <code>derwin</code> , <code>mvderwin</code> , <code>dupwin</code> , <code>wsyncup</code> , <code>syncok</code> , <code>wcursyncup</code> , <code>wsyncdown</code> – create curses windows |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> WINDOW *newwin(int nlines, int ncols, int begin_y, int begin_x); int delwin(WINDOW *win); int mvwin(WINDOW *win, int y, int x); WINDOW *subwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); WINDOW *derwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); int mvderwin(WINDOW *win, int par_y, int par_x); WINDOW *dupwin(WINDOW *win); void wsyncup(WINDOW *win); int syncok(WINDOW *win, bool bf); void wcursyncup(WINDOW *win); void wsyncdown(WINDOW *win);</pre> |
| DESCRIPTION | <p>The <code>newwin()</code> routine creates and returns a pointer to a new window with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. The upper left-hand corner of the window is at line <i>begin_y</i>, column <i>begin_x</i>. If either <i>nlines</i> or <i>ncols</i> is zero, they default to <code>LINES</code> — <i>begin_y</i> and <code>COLS</code> — <i>begin_x</i>. A new full-screen window is created by calling <code>newwin(0, 0, 0, 0)</code>.</p> <p>The <code>delwin()</code> routine deletes the named window, freeing all memory associated with it. Subwindows must be deleted before the main window can be deleted.</p> <p>The <code>mvwin()</code> routine moves the window so that the upper left-hand corner is at position (<i>x</i>, <i>y</i>). If the move would cause the window to be off the screen, it is an error and the window is not moved. Moving subwindows is allowed, but should be avoided.</p> <p>The <code>subwin()</code> routine creates and returns a pointer to a new window with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. The window is at position (<i>begin_y</i>, <i>begin_x</i>) on the screen. (This position is relative to the screen, and not to the window <i>orig</i>.) The window is made in the middle of the window <i>orig</i>, so that changes made to one window will affect both windows. The subwindow shares memory with the window <i>orig</i>. When using this routine, it is necessary to call <code>touchwin()</code> or <code>touchline()</code> on <i>orig</i> before calling <code>wrefresh()</code> on the subwindow.</p> <p>The <code>derwin()</code> routine is the same as <code>subwin()</code>, except that <i>begin_y</i> and <i>begin_x</i> are relative to the origin of the window <i>orig</i> rather than the screen. There is no difference between the subwindows and the derived windows.</p> |

syncok(3CURSES)

The `mvderwin()` routine moves a derived window (or subwindow) inside its parent window. The screen-relative parameters of the window are not changed. This routine is used to display different parts of the parent window at the same physical position on the screen.

The `dupwin()` routine creates an exact duplicate of the window *win*.

Each `curses` window maintains two data structures: the character image structure and the status structure. The character image structure is shared among all windows in the window hierarchy (that is, the window with all subwindows). The status structure, which contains information about individual line changes in the window, is private to each window. The routine `wrefresh()` uses the status data structure when performing screen updating. Since status structures are not shared, changes made to one window in the hierarchy may not be properly reflected on the screen.

The routine `wsyncup()` causes the changes in the status structure of a window to be reflected in the status structures of its ancestors. If `syncok()` is called with second argument `TRUE` then `wsyncup()` is called automatically whenever there is a change in the window.

The routine `wcursyncup()` updates the current cursor position of all the ancestors of the window to reflect the current cursor position of the window.

The routine `wsyncdown()` updates the status structure of the window to reflect the changes in the status structures of its ancestors. Applications seldom call this routine because it is called automatically by `wrefresh()`.

RETURN VALUES

Routines that return an integer return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`delwin()` returns the integer `ERR` upon failure and `OK` upon successful completion.

Routines that return pointers return `NULL` on error.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_refresh(3CURSES)`, `curs_touch(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

If many small changes are made to the window, the `wsyncup()` option could degrade performance.

`syncok(3CURSES)`

Note that `syncok()` may be a macro.

| NAME | syncok, wcuryncup, wsyncdown, wsyncup – synchronize window with its parents or children | | | | | | |
|----------------------|--|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int syncok(WINDOW *win, bool bf); void wcuryncup(WINDOW *win); void wsyncdown(WINDOW *win); void wsyncup(WINDOW *win);</pre> | | | | | | |
| PARAMETERS | <p><i>win</i> Is a pointer to a window.</p> <p><i>bf</i> Is a Boolean expression.</p> | | | | | | |
| DESCRIPTION | <p>The <code>syncok()</code> function uses the value of <i>bf</i> to determine whether or not the window <i>win</i>'s ancestors are implicitly touched whenever there is a change to <i>win</i>. If <i>bf</i> is TRUE, this touching occurs. If <i>bf</i> is FALSE, it does not occur. The initial value for <i>bf</i> is FALSE.</p> <p>The <code>wcuryncup()</code> function moves the cursor in <i>win</i>'s ancestors to match its position in <i>win</i>.</p> <p>The <code>wsyncdown()</code> function touches <i>win</i> if any of its ancestors have been touched.</p> <p>The <code>wsyncup()</code> function touches all ancestors of <i>win</i>.</p> | | | | | | |
| RETURN VALUES | <p>On success, the <code>syncok()</code> function returns OK. Otherwise, it returns ERR.</p> <p>The other functions do not return a value.</p> | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>derwin(3XCURSES)</code> , <code>doupdate(3XCURSES)</code> , <code>is_linetouched(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

termattrs(3CURSES)

| | |
|----------------------|---|
| NAME | <code>curs_termattrs</code> , <code>baudrate</code> , <code>erasechar</code> , <code>has_ic</code> , <code>has_il</code> , <code>killchar</code> , <code>longname</code> , <code>termattrs</code> , <code>termname</code> – curses environment query routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int baudrate(void); char erasechar(void); int has_ic(void); int has_il(void); char killchar(void); char *longname(void); cctype termattrs(void); char *termname(void);</pre> |
| DESCRIPTION | <p>The <code>baudrate()</code> routine returns the output speed of the terminal. The number returned is in bits per second, for example 9600, and is an integer.</p> <p>With the <code>erasechar()</code> routine, the user's current erase character is returned.</p> <p>The <code>has_ic()</code> routine is true if the terminal has insert- and delete-character capabilities.</p> <p>The <code>has_il()</code> routine is true if the terminal has insert- and delete-line capabilities, or can simulate them using scrolling regions. This might be used to determine if it would be appropriate to turn on physical scrolling using <code>scrollok()</code>.</p> <p>With the <code>killchar()</code> routine, the user's current line kill character is returned.</p> <p>The <code>longname()</code> routine returns a pointer to a static area containing a verbose description of the current terminal. The maximum length of a verbose description is 128 characters. It is defined only after the call to <code>initscr()</code> or <code>newterm()</code>. The area is overwritten by each call to <code>newterm()</code> and is not restored by <code>set_term()</code>, so the value should be saved between calls to <code>newterm()</code> if <code>longname()</code> is going to be used with multiple terminals.</p> <p>If a given terminal doesn't support a video attribute that an application program is trying to use, <code>curses</code> may substitute a different video attribute for it. The <code>termattrs()</code> function returns a logical OR of all video attributes supported by the terminal. This information is useful when a <code>curses</code> program needs complete control over the appearance of the screen.</p> <p>The <code>termname()</code> routine returns the value of the environment variable <code>TERM</code> (truncated to 14 characters).</p> |
| RETURN VALUES | <code>longname()</code> and <code>termname()</code> return <code>NULL</code> on error. |

termattrs(3CURSES)

Routines that return an integer return ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_initscr(3CURSES), curs_outopts(3CURSES), curses(3CURSES), attributes(5)

NOTES The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.

Note that termattrs () may be a macro.

term_attrs(3XCURSES)

| NAME | termattrs, term_attrs – get supported terminal video attributes | | | | | | |
|----------------------|--|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> chtype termattrs(void); attr_t term_attrs(void);</pre> | | | | | | |
| DESCRIPTION | <p>The termattrs() function extracts the video attributes of the current terminal which is supported by the chtype data type.</p> <p>The term_attrs() function extracts information for the video attributes of the current terminal which is supported for a cchar_t.</p> | | | | | | |
| RETURN VALUES | <p>The termattrs() function returns a logical OR of A_ values of all video attributes supported by the terminal.</p> <p>The term_attrs() function returns a logical OR of WA_ values of all video attributes supported by the terminal.</p> | | | | | | |
| ERRORS | No errors are defined. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Standard</td></tr><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | attr_get(3XCURSES), attroff(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5) | | | | | | |

| NAME | termattrs, term_attrs – get supported terminal video attributes | | | | | | |
|----------------------|--|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> chtype termattrs(void); attr_t term_attrs(void);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>termattrs()</code> function extracts the video attributes of the current terminal which is supported by the <code>chtype</code> data type.</p> <p>The <code>term_attrs()</code> function extracts information for the video attributes of the current terminal which is supported for a <code>cchar_t</code>.</p> | | | | | | |
| RETURN VALUES | <p>The <code>termattrs()</code> function returns a logical OR of <code>A_</code> values of all video attributes supported by the terminal.</p> <p>The <code>term_attrs()</code> function returns a logical OR of <code>WA_</code> values of all video attributes supported by the terminal.</p> | | | | | | |
| ERRORS | No errors are defined. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>attr_get(3XCURSES)</code> , <code>attroff(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

termname(3CURSES)

| | |
|----------------------|---|
| NAME | <code>curs_termattrs</code> , <code>baudrate</code> , <code>erasechar</code> , <code>has_ic</code> , <code>has_il</code> , <code>killchar</code> , <code>longname</code> , <code>termattrs</code> , <code>termname</code> – curses environment query routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int baudrate(void); char erasechar(void); int has_ic(void); int has_il(void); char killchar(void); char *longname(void); cctype termattrs(void); char *termname(void);</pre> |
| DESCRIPTION | <p>The <code>baudrate()</code> routine returns the output speed of the terminal. The number returned is in bits per second, for example 9600, and is an integer.</p> <p>With the <code>erasechar()</code> routine, the user's current erase character is returned.</p> <p>The <code>has_ic()</code> routine is true if the terminal has insert- and delete-character capabilities.</p> <p>The <code>has_il()</code> routine is true if the terminal has insert- and delete-line capabilities, or can simulate them using scrolling regions. This might be used to determine if it would be appropriate to turn on physical scrolling using <code>scrollok()</code>.</p> <p>With the <code>killchar()</code> routine, the user's current line kill character is returned.</p> <p>The <code>longname()</code> routine returns a pointer to a static area containing a verbose description of the current terminal. The maximum length of a verbose description is 128 characters. It is defined only after the call to <code>initscr()</code> or <code>newterm()</code>. The area is overwritten by each call to <code>newterm()</code> and is not restored by <code>set_term()</code>, so the value should be saved between calls to <code>newterm()</code> if <code>longname()</code> is going to be used with multiple terminals.</p> <p>If a given terminal doesn't support a video attribute that an application program is trying to use, <code>curses</code> may substitute a different video attribute for it. The <code>termattrs()</code> function returns a logical OR of all video attributes supported by the terminal. This information is useful when a <code>curses</code> program needs complete control over the appearance of the screen.</p> <p>The <code>termname()</code> routine returns the value of the environment variable <code>TERM</code> (truncated to 14 characters).</p> |
| RETURN VALUES | <code>longname()</code> and <code>termname()</code> return <code>NULL</code> on error. |

termname(3CURSES)

Routines that return an integer return ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_initscr(3CURSES), curs_outopts(3CURSES), curses(3CURSES), attributes(5)

NOTES The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.

Note that termattrs() may be a macro.

termname(3XCURSES)

NAME | termname – return the value of the environmental variable TERM

SYNOPSIS | `cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \`
| `-R /usr/xpg4/lib -lcurses [library...]`

| `c89 [flag...] file... -lcurses [library...]`

| `#include <curses.h>`

| `char *termname(void);`

DESCRIPTION | The `termname()` function returns a pointer to the value of the environmental variable TERM (truncated to 14 characters).

RETURN VALUES | The `termname()` returns a pointer to the terminal's name.

ERRORS | None.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | `del_curterm(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_termcap, tgetent, tgetflag, tgetnum, tgetstr, tgoto, tputs – curses interfaces (emulated) to the termcap library | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> #include <term.h> int tgetent(char *bp, char *name); int tgetflag(char id[2]); int tgetnum(char id[2]); char *tgetstr(char id[2], char **area); char *tgoto(char *cap, int col, int row); int tputs(char *str, int affcnt, int (*putc)(void));</pre> | | | | |
| DESCRIPTION | <p>These routines are included as a conversion aid for programs that use the <i>termcap</i> library. Their parameters are the same and the routines are emulated using the <i>terminfo</i> database. These routines are supported at Level 2 and should not be used in new applications.</p> <p>The <code>tgetent()</code> routine looks up the termcap entry for <i>name</i>. The emulation ignores the buffer pointer <i>bp</i>.</p> <p>The <code>tgetflag()</code> routine gets the boolean entry for <i>id</i>.</p> <p>The <code>tgetnum()</code> routine gets the numeric entry for <i>id</i>.</p> <p>The <code>tgetstr()</code> routine returns the string entry for <i>id</i>. Use <code>tputs()</code> to output the returned string.</p> <p>The <code>tgoto()</code> routine instantiates the parameters into the given capability. The output from this routine is to be passed to <code>tputs()</code>.</p> <p>The <code>tputs()</code> routine is described on the <code>curs_terminfo(3CURSES)</code> manual page.</p> | | | | |
| RETURN VALUES | <p>Routines that return an integer return ERR upon failure and an integer value other than ERR upon successful completion.</p> <p>Routines that return pointers return NULL on error.</p> | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_terminfo(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>putc(3C)</code> , <code>attributes(5)</code> | | | | |

tgetent(3CURSES)

NOTES | The header `< curses . h >` automatically includes the headers `<stdio . h >` and `<unctrl . h >`.

| | |
|----------------------|---|
| NAME | tgetent, tgetflag, tgetnum, tgetstr, tgoto – emulate the termcap database |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <term.h> int tgetent(char *bp, const char *name); int tgetflag(char id[2]); int tgetnum(char id[2]); char *tgetstr(char id[2], char **area); char *tgoto(char *cap, int col, int row);</pre> |
| PARAMETERS | <p><i>bp</i> Is a pointer to a buffer. This parameter is ignored.</p> <p><i>name</i> Is the termcap entry to look up.</p> <p><i>cap</i> Is the pointer to a termcap capability.</p> <p><i>area</i> Is a pointer to the area where tgetstr() stores the decoded string.</p> <p><i>col</i> Is the column placement of the new cursor.</p> <p><i>row</i> Is the row placement of the new cursor.</p> |
| DESCRIPTION | <p>The tgetent() function looks up the termcap entry for <i>name</i>. The emulation ignores the buffer pointer <i>bp</i>.</p> <p>The tgetflag() function gets the Boolean entry for <i>id</i>.</p> <p>The tgetnum() function gets the numeric entry for <i>id</i>.</p> <p>The tgetstr() function gets the string entry for <i>id</i>. If <i>area</i> is not a null pointer and does not point to a null pointer, tgetstr() copies the string entry into the buffer pointed to by <i>*area</i> and advances the variable pointed to by <i>area</i> to the first byte after the copy of the string entry.</p> <p>The tgoto() function instantiates the parameters <i>col</i> and <i>row</i> into the capability <i>cap</i> and returns a pointer to the resulting string.</p> <p>All of the information available in the terminfo database need not be available through these functions.</p> |
| RETURN VALUES | <p>Upon successful completion, those functions that return integers return OK. Otherwise, they return ERR.</p> <p>Those functions that return pointers return a null pointer when an error occurs.</p> |
| ERRORS | No errors are defined. |

tgetent(3XCURSES)

USAGE | These functions are included as a conversion aid for programs that use the `termcap` library. Their arguments are the same and the functions are emulated using the `terminfo` database.

These functions are only guaranteed to operate reliably on character sets in which each character fits into a single byte, whose attributes can be expressed using only constants with the `A_` prefix.

Any terminal capabilities from the `terminfo` database that cannot be retrieved using these functions can be retrieved using the functions described on the `tigetflag(3XCURSES)` manual page.

Portable applications must use `tputs(3XCURSES)` to output the strings returned by `tgetstr()` and `tgoto()`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | `libcurses(3XCURSES)`, `putp(3XCURSES)`, `setupterm(3XCURSES)`, `tigetflag(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_termcap, tgetent, tgetflag, tgetnum, tgetstr, tgoto, tputs – curses interfaces (emulated) to the termcap library | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> #include <term.h> int tgetent(char *bp, char *name); int tgetflag(char id[2]); int tgetnum(char id[2]); char *tgetstr(char id[2], char **area); char *tgoto(char *cap, int col, int row); int tputs(char *str, int affcnt, int (*putc)(void));</pre> | | | | |
| DESCRIPTION | <p>These routines are included as a conversion aid for programs that use the <i>termcap</i> library. Their parameters are the same and the routines are emulated using the <i>terminfo</i> database. These routines are supported at Level 2 and should not be used in new applications.</p> <p>The <code>tgetent()</code> routine looks up the termcap entry for <i>name</i>. The emulation ignores the buffer pointer <i>bp</i>.</p> <p>The <code>tgetflag()</code> routine gets the boolean entry for <i>id</i>.</p> <p>The <code>tgetnum()</code> routine gets the numeric entry for <i>id</i>.</p> <p>The <code>tgetstr()</code> routine returns the string entry for <i>id</i>. Use <code>tputs()</code> to output the returned string.</p> <p>The <code>tgoto()</code> routine instantiates the parameters into the given capability. The output from this routine is to be passed to <code>tputs()</code>.</p> <p>The <code>tputs()</code> routine is described on the <code>curs_terminfo(3CURSES)</code> manual page.</p> | | | | |
| RETURN VALUES | <p>Routines that return an integer return ERR upon failure and an integer value other than ERR upon successful completion.</p> <p>Routines that return pointers return NULL on error.</p> | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_terminfo(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>putc(3C)</code> , <code>attributes(5)</code> | | | | |

tgetflag(3CURSES)

NOTES | The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

| | |
|----------------------|---|
| NAME | tgetent, tgetflag, tgetnum, tgetstr, tgoto – emulate the termcap database |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <term.h> int tgetent(char *bp, const char *name); int tgetflag(char id[2]); int tgetnum(char id[2]); char *tgetstr(char id[2], char **area); char *tgoto(char *cap, int col, int row);</pre> |
| PARAMETERS | <p><i>bp</i> Is a pointer to a buffer. This parameter is ignored.</p> <p><i>name</i> Is the termcap entry to look up.</p> <p><i>cap</i> Is the pointer to a termcap capability.</p> <p><i>area</i> Is a pointer to the area where tgetstr() stores the decoded string.</p> <p><i>col</i> Is the column placement of the new cursor.</p> <p><i>row</i> Is the row placement of the new cursor.</p> |
| DESCRIPTION | <p>The tgetent() function looks up the termcap entry for <i>name</i>. The emulation ignores the buffer pointer <i>bp</i>.</p> <p>The tgetflag() function gets the Boolean entry for <i>id</i>.</p> <p>The tgetnum() function gets the numeric entry for <i>id</i>.</p> <p>The tgetstr() function gets the string entry for <i>id</i>. If <i>area</i> is not a null pointer and does not point to a null pointer, tgetstr() copies the string entry into the buffer pointed to by <i>*area</i> and advances the variable pointed to by <i>area</i> to the first byte after the copy of the string entry.</p> <p>The tgoto() function instantiates the parameters <i>col</i> and <i>row</i> into the capability <i>cap</i> and returns a pointer to the resulting string.</p> <p>All of the information available in the terminfo database need not be available through these functions.</p> |
| RETURN VALUES | <p>Upon successful completion, those functions that return integers return OK. Otherwise, they return ERR.</p> <p>Those functions that return pointers return a null pointer when an error occurs.</p> |
| ERRORS | No errors are defined. |

tgetflag(3XCURSES)

USAGE | These functions are included as a conversion aid for programs that use the `termcap` library. Their arguments are the same and the functions are emulated using the `terminfo` database.

These functions are only guaranteed to operate reliably on character sets in which each character fits into a single byte, whose attributes can be expressed using only constants with the `A_` prefix.

Any terminal capabilities from the `terminfo` database that cannot be retrieved using these functions can be retrieved using the functions described on the `tigetflag(3XCURSES)` manual page.

Portable applications must use `tputs(3XCURSES)` to output the strings returned by `tgetstr()` and `tgoto()`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | `libcurses(3XCURSES)`, `putp(3XCURSES)`, `setupterm(3XCURSES)`, `tigetflag(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_termcap, tgetent, tgetflag, tgetnum, tgetstr, tgoto, tputs – curses interfaces (emulated) to the termcap library | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> #include <term.h> int tgetent(char *bp, char *name); int tgetflag(char id[2]); int tgetnum(char id[2]); char *tgetstr(char id[2], char **area); char *tgoto(char *cap, int col, int row); int tputs(char *str, int affcnt, int (*putc)(void));</pre> | | | | |
| DESCRIPTION | <p>These routines are included as a conversion aid for programs that use the <i>termcap</i> library. Their parameters are the same and the routines are emulated using the <i>terminfo</i> database. These routines are supported at Level 2 and should not be used in new applications.</p> <p>The <code>tgetent()</code> routine looks up the termcap entry for <i>name</i>. The emulation ignores the buffer pointer <i>bp</i>.</p> <p>The <code>tgetflag()</code> routine gets the boolean entry for <i>id</i>.</p> <p>The <code>tgetnum()</code> routine gets the numeric entry for <i>id</i>.</p> <p>The <code>tgetstr()</code> routine returns the string entry for <i>id</i>. Use <code>tputs()</code> to output the returned string.</p> <p>The <code>tgoto()</code> routine instantiates the parameters into the given capability. The output from this routine is to be passed to <code>tputs()</code>.</p> <p>The <code>tputs()</code> routine is described on the <code>curs_terminfo(3CURSES)</code> manual page.</p> | | | | |
| RETURN VALUES | <p>Routines that return an integer return ERR upon failure and an integer value other than ERR upon successful completion.</p> <p>Routines that return pointers return NULL on error.</p> | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_terminfo(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>putc(3C)</code> , <code>attributes(5)</code> | | | | |

tgetnum(3CURSES)

NOTES | The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

| | |
|----------------------|---|
| NAME | tgetent, tgetflag, tgetnum, tgetstr, tgoto – emulate the termcap database |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <term.h> int tgetent(char *bp, const char *name); int tgetflag(char id[2]); int tgetnum(char id[2]); char *tgetstr(char id[2], char **area); char *tgoto(char *cap, int col, int row);</pre> |
| PARAMETERS | <p><i>bp</i> Is a pointer to a buffer. This parameter is ignored.</p> <p><i>name</i> Is the termcap entry to look up.</p> <p><i>cap</i> Is the pointer to a termcap capability.</p> <p><i>area</i> Is a pointer to the area where tgetstr() stores the decoded string.</p> <p><i>col</i> Is the column placement of the new cursor.</p> <p><i>row</i> Is the row placement of the new cursor.</p> |
| DESCRIPTION | <p>The tgetent() function looks up the termcap entry for <i>name</i>. The emulation ignores the buffer pointer <i>bp</i>.</p> <p>The tgetflag() function gets the Boolean entry for <i>id</i>.</p> <p>The tgetnum() function gets the numeric entry for <i>id</i>.</p> <p>The tgetstr() function gets the string entry for <i>id</i>. If <i>area</i> is not a null pointer and does not point to a null pointer, tgetstr() copies the string entry into the buffer pointed to by <i>*area</i> and advances the variable pointed to by <i>area</i> to the first byte after the copy of the string entry.</p> <p>The tgoto() function instantiates the parameters <i>col</i> and <i>row</i> into the capability <i>cap</i> and returns a pointer to the resulting string.</p> <p>All of the information available in the terminfo database need not be available through these functions.</p> |
| RETURN VALUES | <p>Upon successful completion, those functions that return integers return OK. Otherwise, they return ERR.</p> <p>Those functions that return pointers return a null pointer when an error occurs.</p> |
| ERRORS | No errors are defined. |

tgetnum(3XCURSES)

USAGE | These functions are included as a conversion aid for programs that use the `termcap` library. Their arguments are the same and the functions are emulated using the `terminfo` database.

These functions are only guaranteed to operate reliably on character sets in which each character fits into a single byte, whose attributes can be expressed using only constants with the `A_` prefix.

Any terminal capabilities from the `terminfo` database that cannot be retrieved using these functions can be retrieved using the functions described on the `tigetflag(3XCURSES)` manual page.

Portable applications must use `tputs(3XCURSES)` to output the strings returned by `tgetstr()` and `tgoto()`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | `libcurses(3XCURSES)`, `putp(3XCURSES)`, `setupterm(3XCURSES)`, `tigetflag(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_termcap, tgetent, tgetflag, tgetnum, tgetstr, tgoto, tputs – curses interfaces (emulated) to the termcap library | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> #include <term.h> int tgetent(char *bp, char *name); int tgetflag(char id[2]); int tgetnum(char id[2]); char *tgetstr(char id[2], char **area); char *tgoto(char *cap, int col, int row); int tputs(char *str, int affcnt, int (*putc)(void));</pre> | | | | |
| DESCRIPTION | <p>These routines are included as a conversion aid for programs that use the <i>termcap</i> library. Their parameters are the same and the routines are emulated using the <i>terminfo</i> database. These routines are supported at Level 2 and should not be used in new applications.</p> <p>The <code>tgetent()</code> routine looks up the termcap entry for <i>name</i>. The emulation ignores the buffer pointer <i>bp</i>.</p> <p>The <code>tgetflag()</code> routine gets the boolean entry for <i>id</i>.</p> <p>The <code>tgetnum()</code> routine gets the numeric entry for <i>id</i>.</p> <p>The <code>tgetstr()</code> routine returns the string entry for <i>id</i>. Use <code>tputs()</code> to output the returned string.</p> <p>The <code>tgoto()</code> routine instantiates the parameters into the given capability. The output from this routine is to be passed to <code>tputs()</code>.</p> <p>The <code>tputs()</code> routine is described on the <code>curs_terminfo(3CURSES)</code> manual page.</p> | | | | |
| RETURN VALUES | <p>Routines that return an integer return <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion.</p> <p>Routines that return pointers return <code>NULL</code> on error.</p> | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_terminfo(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>putc(3C)</code> , <code>attributes(5)</code> | | | | |

tgetstr(3CURSES)

NOTES | The header `< curses . h >` automatically includes the headers `<stdio . h >` and `<unctrl . h >`.

| | |
|----------------------|---|
| NAME | tgetent, tgetflag, tgetnum, tgetstr, tgoto – emulate the termcap database |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <term.h> int tgetent(char *bp, const char *name); int tgetflag(char id[2]); int tgetnum(char id[2]); char *tgetstr(char id[2], char **area); char *tgoto(char *cap, int col, int row);</pre> |
| PARAMETERS | <p><i>bp</i> Is a pointer to a buffer. This parameter is ignored.</p> <p><i>name</i> Is the termcap entry to look up.</p> <p><i>cap</i> Is the pointer to a termcap capability.</p> <p><i>area</i> Is a pointer to the area where tgetstr() stores the decoded string.</p> <p><i>col</i> Is the column placement of the new cursor.</p> <p><i>row</i> Is the row placement of the new cursor.</p> |
| DESCRIPTION | <p>The tgetent() function looks up the termcap entry for <i>name</i>. The emulation ignores the buffer pointer <i>bp</i>.</p> <p>The tgetflag() function gets the Boolean entry for <i>id</i>.</p> <p>The tgetnum() function gets the numeric entry for <i>id</i>.</p> <p>The tgetstr() function gets the string entry for <i>id</i>. If <i>area</i> is not a null pointer and does not point to a null pointer, tgetstr() copies the string entry into the buffer pointed to by <i>*area</i> and advances the variable pointed to by <i>area</i> to the first byte after the copy of the string entry.</p> <p>The tgoto() function instantiates the parameters <i>col</i> and <i>row</i> into the capability <i>cap</i> and returns a pointer to the resulting string.</p> <p>All of the information available in the terminfo database need not be available through these functions.</p> |
| RETURN VALUES | <p>Upon successful completion, those functions that return integers return OK. Otherwise, they return ERR.</p> <p>Those functions that return pointers return a null pointer when an error occurs.</p> |
| ERRORS | No errors are defined. |

tgetstr(3XCURSES)

USAGE These functions are included as a conversion aid for programs that use the `termcap` library. Their arguments are the same and the functions are emulated using the `terminfo` database.

These functions are only guaranteed to operate reliably on character sets in which each character fits into a single byte, whose attributes can be expressed using only constants with the `A_` prefix.

Any terminal capabilities from the `terminfo` database that cannot be retrieved using these functions can be retrieved using the functions described on the `tigetflag(3XCURSES)` manual page.

Portable applications must use `tputs(3XCURSES)` to output the strings returned by `tgetstr()` and `tgoto()`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `libcurses(3XCURSES)`, `putp(3XCURSES)`, `setupterm(3XCURSES)`, `tigetflag(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_termcap, tgetent, tgetflag, tgetnum, tgetstr, tgoto, tputs – curses interfaces (emulated) to the termcap library | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> #include <term.h> int tgetent(char *bp, char *name); int tgetflag(char id[2]); int tgetnum(char id[2]); char *tgetstr(char id[2], char **area); char *tgoto(char *cap, int col, int row); int tputs(char *str, int affcnt, int (*putc)(void));</pre> | | | | |
| DESCRIPTION | <p>These routines are included as a conversion aid for programs that use the <i>termcap</i> library. Their parameters are the same and the routines are emulated using the <i>terminfo</i> database. These routines are supported at Level 2 and should not be used in new applications.</p> <p>The <code>tgetent()</code> routine looks up the termcap entry for <i>name</i>. The emulation ignores the buffer pointer <i>bp</i>.</p> <p>The <code>tgetflag()</code> routine gets the boolean entry for <i>id</i>.</p> <p>The <code>tgetnum()</code> routine gets the numeric entry for <i>id</i>.</p> <p>The <code>tgetstr()</code> routine returns the string entry for <i>id</i>. Use <code>tputs()</code> to output the returned string.</p> <p>The <code>tgoto()</code> routine instantiates the parameters into the given capability. The output from this routine is to be passed to <code>tputs()</code>.</p> <p>The <code>tputs()</code> routine is described on the <code>curs_terminfo(3CURSES)</code> manual page.</p> | | | | |
| RETURN VALUES | <p>Routines that return an integer return ERR upon failure and an integer value other than ERR upon successful completion.</p> <p>Routines that return pointers return NULL on error.</p> | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_terminfo(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>putc(3C)</code> , <code>attributes(5)</code> | | | | |

tgoto(3CURSES)

NOTES | The header `< curses . h >` automatically includes the headers `<stdio . h >` and `<unctrl . h >`.

| | |
|----------------------|---|
| NAME | tgetent, tgetflag, tgetnum, tgetstr, tgoto – emulate the termcap database |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <term.h> int tgetent(char *bp, const char *name); int tgetflag(char id[2]); int tgetnum(char id[2]); char *tgetstr(char id[2], char **area); char *tgoto(char *cap, int col, int row);</pre> |
| PARAMETERS | <p><i>bp</i> Is a pointer to a buffer. This parameter is ignored.</p> <p><i>name</i> Is the termcap entry to look up.</p> <p><i>cap</i> Is the pointer to a termcap capability.</p> <p><i>area</i> Is a pointer to the area where tgetstr() stores the decoded string.</p> <p><i>col</i> Is the column placement of the new cursor.</p> <p><i>row</i> Is the row placement of the new cursor.</p> |
| DESCRIPTION | <p>The tgetent() function looks up the termcap entry for <i>name</i>. The emulation ignores the buffer pointer <i>bp</i>.</p> <p>The tgetflag() function gets the Boolean entry for <i>id</i>.</p> <p>The tgetnum() function gets the numeric entry for <i>id</i>.</p> <p>The tgetstr() function gets the string entry for <i>id</i>. If <i>area</i> is not a null pointer and does not point to a null pointer, tgetstr() copies the string entry into the buffer pointed to by <i>*area</i> and advances the variable pointed to by <i>area</i> to the first byte after the copy of the string entry.</p> <p>The tgoto() function instantiates the parameters <i>col</i> and <i>row</i> into the capability <i>cap</i> and returns a pointer to the resulting string.</p> <p>All of the information available in the terminfo database need not be available through these functions.</p> |
| RETURN VALUES | <p>Upon successful completion, those functions that return integers return OK. Otherwise, they return ERR.</p> <p>Those functions that return pointers return a null pointer when an error occurs.</p> |
| ERRORS | No errors are defined. |

tgoto(3XCURSES)

USAGE These functions are included as a conversion aid for programs that use the `termcap` library. Their arguments are the same and the functions are emulated using the `terminfo` database.

These functions are only guaranteed to operate reliably on character sets in which each character fits into a single byte, whose attributes can be expressed using only constants with the `A_` prefix.

Any terminal capabilities from the `terminfo` database that cannot be retrieved using these functions can be retrieved using the functions described on the `tigetflag(3XCURSES)` manual page.

Portable applications must use `tputs(3XCURSES)` to output the strings returned by `tgetstr()` and `tgoto()`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `libcurses(3XCURSES)`, `putp(3XCURSES)`, `setupterm(3XCURSES)`, `tigetflag(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|---|
| NAME | curs_terminfo, setupterm, setterm, set_curterm, del_curterm, restartterm, tparm, tputs, putp, vidputs, vidattr, mvcur, tigetflag, tigetnum, tigetstr – curses interfaces to terminfo database |
| SYNOPSIS | <pre> cc [flag ...] file ... -lcurses [library ...] #include <curses.h> #include <term.h> int setupterm(char *term, int fildes, int *errret); int setterm(char *term); int set_curterm(TERMINAL *nterm); int del_curterm(TERMINAL *oterm); int restartterm(char *term, int fildes, int *errret); char *tparm(char *str, long int p1, long int p2, long int p3, long int p4, long int p5, long int p6, long int p7, long int p8, long int p9); int tputs(char *str, int affcnt, int (*putc)(char)); int putp(char *str); int vidputs(chtype attrs, int (*putc)(char)); int vidattr(chtype attrs); int mvcur(int oldrow, int oldcol, int newrow, int newcol); int tigetflag(char *capname); int tigetnum(char *capname); char *tigetstr(char *capname); </pre> |
| DESCRIPTION | <p>These low-level routines must be called by programs that have to deal directly with the <i>terminfo</i> database to handle certain terminal capabilities, such as programming function keys. For all other functionality, <i>curses</i> routines are more suitable and their use is recommended.</p> <p>Initially, <code>setupterm()</code> should be called. Note that <code>setupterm()</code> is automatically called by <code>initscr()</code> and <code>newterm()</code>. This defines the set of terminal-dependent variables (listed in <code>terminfo(4)</code>). The <i>terminfo</i> variables <code>lines</code> and <code>columns</code> are initialized by <code>setupterm()</code> as follows: If <code>use_env(FALSE)</code> has been called, values for <code>lines</code> and <code>columns</code> specified in <i>terminfo</i> are used. Otherwise, if the environment variables <code>LINES</code> and <code>COLUMNS</code> exist, their values are used. If these environment variables do not exist and the program is running in a window, the current window size is used. Otherwise, if the environment variables do not exist, the values for <code>lines</code> and <code>columns</code> specified in the <i>terminfo</i> database are used.</p> <p>The headers <code><curses.h></code> and <code><term.h></code> should be included (in this order) to get the definitions for these strings, numbers, and flags. Parameterized strings should be passed through <code>tparm()</code> to instantiate them. All <i>terminfo</i> strings (including the output</p> |

tigetflag(3CURSES)

of `tparm()` should be printed with `tputs()` or `putp()`. Call the `reset_shell_mode()` routine to restore the tty modes before exiting (see `curs_kernel(3CURSES)`). Programs which use cursor addressing should output `enter_ca_mode` upon startup and should output `exit_ca_mode` before exiting. Programs desiring shell escapes should call `reset_shell_mode` and output `exit_ca_mode` before the shell is called and should output `enter_ca_mode` and call `reset_prog_mode` after returning from the shell.

The `setupterm()` routine reads in the *terminfo* database, initializing the *terminfo* structures, but does not set up the output virtualization structures used by *curses*. The terminal type is the character string *term*; if *term* is null, the environment variable `TERM` is used. All output is to file descriptor *fildev* which is initialized for output. If *errret* is not null, then `setupterm()` returns `OK` or `ERR` and stores a status value in the integer pointed to by *errret*. A status of 1 in *errret* is normal, 0 means that the terminal could not be found, and -1 means that the *terminfo* database could not be found. If *errret* is null, `setupterm()` prints an error message upon finding an error and exits. Thus, the simplest call is:

```
setupterm((char *)0, 1, (int *)0);
```

which uses all the defaults and sends the output to `stdout`.

The `setterm()` routine is being replaced by `setupterm()`. The call:

```
setupterm(term, 1, (int *)0)
```

provides the same functionality as `setterm(term)`. The `setterm()` routine is included here for compatibility and is supported at Level 2.

The `set_curterm()` routine sets the variable `cur_term` to *nterm*, and makes all of the *terminfo* boolean, numeric, and string variables use the values from *nterm*.

The `del_curterm()` routine frees the space pointed to by *oterm* and makes it available for further use. If *oterm* is the same as `cur_term`, references to any of the *terminfo* boolean, numeric, and string variables thereafter may refer to invalid memory locations until another `setupterm()` has been called.

The `restartterm()` routine is similar to `setupterm()` and `initscr()`, except that it is called after restoring memory to a previous state. It assumes that the windows and the input and output options are the same as when memory was saved, but the terminal type and baud rate may be different.

The `tparm()` routine instantiates the string *str* with parameters *pi*. A pointer is returned to the result of *str* with the parameters applied.

The `tputs()` routine applies padding information to the string *str* and outputs it. The *str* must be a *terminfo* string variable or the return value from `tparm()`, `tgetstr()`, or `tgoto()`. *affcnt* is the number of lines affected, or 1 if not applicable. *putc* is a `putchar()`-like routine to which the characters are passed, one at a time.

tigetflag(3CURSES)

The `putp()` routine calls `tputs(str, 1, putchar)`. Note that the output of `putpA()` always goes to `stdout`, not to the *files* specified in `setupterm()`.

The `vidputs()` routine displays the string on the terminal in the video attribute mode *attrs*, which is any combination of the attributes listed in `curses(3CURSES)`. The characters are passed to the `putchar()`-like routine `putc()`.

The `vidattr()` routine is like the `vidputs()` routine, except that it outputs through `putchar()`.

The `mvcur()` routine provides low-level cursor motion.

The `tigetflag()`, `tigetnum()` and `tigetstr()` routines return the value of the capability corresponding to the *terminfo capname* passed to them, such as `xenl`.

With the `tigetflag()` routine, the value `-1` is returned if *capname* is not a boolean capability.

With the `tigetnum()` routine, the value `-2` is returned if *capname* is not a numeric capability.

With the `tigetstr()` routine, the value `(char *)-1` is returned if *capname* is not a string capability.

The *capname* for each capability is given in the table column entitled *capname* code in the capabilities section of `terminfo(4)`.

```
char *boolnames, *boolcodes, *boolfnames
char *numnames, *numcodes, *numfnames
char *strnames, *strcodes, *strfnames
```

These null-terminated arrays contain the *capnames*, the *termcap* codes, and the full C names, for each of the *terminfo* variables.

RETURN VALUES

All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

Routines that return pointers always return `NULL` on error.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_initscr(3CURSES)`, `curs_kernel(3CURSES)`, `curs_termcap(3CURSES)`, `curses(3CURSES)`, `putc(3C)`, `terminfo(4)`, `attributes(5)`

tigetflag(3CURSES)

NOTES | The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

The `setupterm()` routine should be used in place of `setterm()`.

Note that `vidattr()` and `vidputs()` may be macros.

| | |
|----------------------|---|
| NAME | tigetflag, tigetnum, tigetstr, tparm – return the value of a terminfo capability |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <term.h> int tigetflag(char *capname); int tigetnum(char *capname); char *tigetstr(char *capname); char *tparm(char *cap, long p1, long p2, long p3, long p4, long p5, long p6, long p7, long p8, long p9);</pre> |
| PARAMETERS | <p><i>capname</i> Is the name of the terminfo capability for which the value is required.</p> <p><i>cap</i> Is a pointer to a string capability.</p> <p><i>p1...p9</i> Are the parameters to be instantiated.</p> |
| DESCRIPTION | <p>The <code>tigetflag()</code>, <code>tigetnum()</code>, and <code>tigetstr()</code> functions return values for terminfo capabilities passed to them.</p> <p>The following null-terminated arrays contain the <i>capnames</i>, the termcap codes and full C names for each of the terminfo variables.</p> <pre>char *boolnames, *boolcodes, *boolfnames char *numnames, *numcodes, *numfnames char *strnames, *strcodes, *strfnames</pre> <p>The <code>tparm()</code> function instantiates a parameterized string using nine arguments. The string is suitable for output processing by <code>tputs()</code>.</p> |
| RETURN VALUES | <p>On success, the <code>tigetflg()</code>, <code>tigetnum()</code>, and <code>tigetstr()</code> functions return the specified terminfo capability.</p> <p><code>tigetflag()</code> returns <code>-1</code> if <i>capname</i> is not a Boolean capability.</p> <p><code>tigetnum()</code> returns <code>-2</code> if <i>capname</i> is not a numeric capability.</p> <p><code>tigetstr()</code> returns <code>(char *)-1</code> if <i>capname</i> is not a string capability.</p> <p>On success, the <code>tparm()</code> function returns <i>cap</i> in a static buffer with the parameterization resolved. Otherwise, it returns a null pointer.</p> |
| ERRORS | None. |

tigetflag(3XCURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `libcurses(3XCURSES)`, `tgetent(3XCURSES)`, `terminfo(4)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | curs_terminfo, setupterm, setterm, set_curterm, del_curterm, restartterm, tparm, tputs, putp, vidputs, vidattr, mvcur, tigetflag, tigetnum, tigetstr – curses interfaces to terminfo database |
| SYNOPSIS | <pre> cc [flag ...] file ... -lcurses [library ...] #include <curses.h> #include <term.h> int setupterm(char *term, int fildes, int *errret); int setterm(char *term); int set_curterm(TERMINAL *nterm); int del_curterm(TERMINAL *oterm); int restartterm(char *term, int fildes, int *errret); char *tparm(char *str, long int p1, long int p2, long int p3, long int p4, long int p5, long int p6, long int p7, long int p8, long int p9); int tputs(char *str, int affcnt, int (*putc)(char)); int putp(char *str); int vidputs(chtype attrs, int (*putc)(char)); int vidattr(chtype attrs); int mvcur(int oldrow, int oldcol, int newrow, int newcol); int tigetflag(char *capname); int tigetnum(char *capname); char *tigetstr(char *capname); </pre> |
| DESCRIPTION | <p>These low-level routines must be called by programs that have to deal directly with the <i>terminfo</i> database to handle certain terminal capabilities, such as programming function keys. For all other functionality, curses routines are more suitable and their use is recommended.</p> <p>Initially, <code>setupterm()</code> should be called. Note that <code>setupterm()</code> is automatically called by <code>initscr()</code> and <code>newterm()</code>. This defines the set of terminal-dependent variables (listed in <code>terminfo(4)</code>). The <i>terminfo</i> variables <code>lines</code> and <code>columns</code> are initialized by <code>setupterm()</code> as follows: If <code>use_env(FALSE)</code> has been called, values for <code>lines</code> and <code>columns</code> specified in <i>terminfo</i> are used. Otherwise, if the environment variables <code>LINES</code> and <code>COLUMNS</code> exist, their values are used. If these environment variables do not exist and the program is running in a window, the current window size is used. Otherwise, if the environment variables do not exist, the values for <code>lines</code> and <code>columns</code> specified in the <i>terminfo</i> database are used.</p> <p>The headers <code><curses.h></code> and <code><term.h></code> should be included (in this order) to get the definitions for these strings, numbers, and flags. Parameterized strings should be passed through <code>tparm()</code> to instantiate them. All <i>terminfo</i> strings (including the output</p> |

tigetnum(3CURSES)

of `tparm()` should be printed with `tputs()` or `putp()`. Call the `reset_shell_mode()` routine to restore the tty modes before exiting (see `curs_kernel(3CURSES)`). Programs which use cursor addressing should output `enter_ca_mode` upon startup and should output `exit_ca_mode` before exiting. Programs desiring shell escapes should call `reset_shell_mode` and output `exit_ca_mode` before the shell is called and should output `enter_ca_mode` and call `reset_prog_mode` after returning from the shell.

The `setupterm()` routine reads in the *terminfo* database, initializing the *terminfo* structures, but does not set up the output virtualization structures used by *curses*. The terminal type is the character string *term*; if *term* is null, the environment variable `TERM` is used. All output is to file descriptor *fildev* which is initialized for output. If *errret* is not null, then `setupterm()` returns `OK` or `ERR` and stores a status value in the integer pointed to by *errret*. A status of 1 in *errret* is normal, 0 means that the terminal could not be found, and -1 means that the *terminfo* database could not be found. If *errret* is null, `setupterm()` prints an error message upon finding an error and exits. Thus, the simplest call is:

```
setupterm((char *)0, 1, (int *)0);
```

which uses all the defaults and sends the output to `stdout`.

The `setterm()` routine is being replaced by `setupterm()`. The call:

```
setupterm(term, 1, (int *)0)
```

provides the same functionality as `setterm(term)`. The `setterm()` routine is included here for compatibility and is supported at Level 2.

The `set_curterm()` routine sets the variable `cur_term` to *nterm*, and makes all of the *terminfo* boolean, numeric, and string variables use the values from *nterm*.

The `del_curterm()` routine frees the space pointed to by *oterm* and makes it available for further use. If *oterm* is the same as `cur_term`, references to any of the *terminfo* boolean, numeric, and string variables thereafter may refer to invalid memory locations until another `setupterm()` has been called.

The `restartterm()` routine is similar to `setupterm()` and `initscr()`, except that it is called after restoring memory to a previous state. It assumes that the windows and the input and output options are the same as when memory was saved, but the terminal type and baud rate may be different.

The `tparm()` routine instantiates the string *str* with parameters *pi*. A pointer is returned to the result of *str* with the parameters applied.

The `tputs()` routine applies padding information to the string *str* and outputs it. The *str* must be a *terminfo* string variable or the return value from `tparm()`, `tgetstr()`, or `tgoto()`. *affcnt* is the number of lines affected, or 1 if not applicable. *putc* is a `putchar()`-like routine to which the characters are passed, one at a time.

tigetnum(3CURSES)

The `putp()` routine calls `tputs(str, 1, putchar)`. Note that the output of `putpA()` always goes to `stdout`, not to the *files* specified in `setupterm()`.

The `vidputs()` routine displays the string on the terminal in the video attribute mode *attrs*, which is any combination of the attributes listed in `curses(3CURSES)`. The characters are passed to the `putchar()`-like routine `putc()`.

The `vidattr()` routine is like the `vidputs()` routine, except that it outputs through `putchar()`.

The `mvcur()` routine provides low-level cursor motion.

The `tigetflag()`, `tigetnum()` and `tigetstr()` routines return the value of the capability corresponding to the *terminfo capname* passed to them, such as `xenl`.

With the `tigetflag()` routine, the value `-1` is returned if *capname* is not a boolean capability.

With the `tigetnum()` routine, the value `-2` is returned if *capname* is not a numeric capability.

With the `tigetstr()` routine, the value `(char *)-1` is returned if *capname* is not a string capability.

The *capname* for each capability is given in the table column entitled *capname* code in the capabilities section of `terminfo(4)`.

```
char *boolnames, *boolcodes, *boolfnames
char *numnames, *numcodes, *numfnames
char *strnames, *strcodes, *strfnames
```

These null-terminated arrays contain the *capnames*, the *termcap* codes, and the full C names, for each of the *terminfo* variables.

RETURN VALUES

All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

Routines that return pointers always return `NULL` on error.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_initscr(3CURSES)`, `curs_kernel(3CURSES)`, `curs_termcap(3CURSES)`, `curses(3CURSES)`, `putc(3C)`, `terminfo(4)`, `attributes(5)`

tigetnum(3CURSES)

NOTES | The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

The `setupterm()` routine should be used in place of `setterm()`.

Note that `vidattr()` and `vidputs()` may be macros.

| | |
|----------------------|---|
| NAME | tigetflag, tigetnum, tigetstr, tparm – return the value of a terminfo capability |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <term.h> int tigetflag(char *capname); int tigetnum(char *capname); char *tigetstr(char *capname); char *tparm(char *cap, long p1, long p2, long p3, long p4, long p5, long p6, long p7, long p8, long p9);</pre> |
| PARAMETERS | <p><i>capname</i> Is the name of the terminfo capability for which the value is required.</p> <p><i>cap</i> Is a pointer to a string capability.</p> <p><i>p1...p9</i> Are the parameters to be instantiated.</p> |
| DESCRIPTION | <p>The <code>tigetflag()</code>, <code>tigetnum()</code>, and <code>tigetstr()</code> functions return values for terminfo capabilities passed to them.</p> <p>The following null-terminated arrays contain the <i>capnames</i>, the termcap codes and full C names for each of the terminfo variables.</p> <pre>char *boolnames, *boolcodes, *boolfnames char *numnames, *numcodes, *numfnames char *strnames, *strcodes, *strfnames</pre> <p>The <code>tparm()</code> function instantiates a parameterized string using nine arguments. The string is suitable for output processing by <code>tputs()</code>.</p> |
| RETURN VALUES | <p>On success, the <code>tigetflg()</code>, <code>tigetnum()</code>, and <code>tigetstr()</code> functions return the specified terminfo capability.</p> <p><code>tigetflag()</code> returns <code>-1</code> if <i>capname</i> is not a Boolean capability.</p> <p><code>tigetnum()</code> returns <code>-2</code> if <i>capname</i> is not a numeric capability.</p> <p><code>tigetstr()</code> returns <code>(char *)-1</code> if <i>capname</i> is not a string capability.</p> <p>On success, the <code>tparm()</code> function returns <i>cap</i> in a static buffer with the parameterization resolved. Otherwise, it returns a null pointer.</p> |
| ERRORS | None. |

tigetnum(3XCURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `libcurses(3XCURSES)`, `tgetent(3XCURSES)`, `terminfo(4)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | curs_terminfo, setupterm, setterm, set_curterm, del_curterm, restartterm, tparm, tputs, putp, vidputs, vidattr, mvcur, tigetflag, tigetnum, tigetstr – curses interfaces to terminfo database |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> #include <term.h> int setupterm(char *term, int fildes, int *errret); int setterm(char *term); int set_curterm(TERMINAL *nterm); int del_curterm(TERMINAL *oterm); int restartterm(char *term, int fildes, int *errret); char *tparm(char *str, long int p1, long int p2, long int p3, long int p4, long int p5, long int p6, long int p7, long int p8, long int p9); int tputs(char *str, int affcnt, int (*putc)(char)); int putp(char *str); int vidputs(chtype attrs, int (*putc)(char)); int vidattr(chtype attrs); int mvcur(int oldrow, int oldcol, int newrow, int newcol); int tigetflag(char *capname); int tigetnum(char *capname); char *tigetstr(char *capname);</pre> |
| DESCRIPTION | <p>These low-level routines must be called by programs that have to deal directly with the <i>terminfo</i> database to handle certain terminal capabilities, such as programming function keys. For all other functionality, curses routines are more suitable and their use is recommended.</p> <p>Initially, <code>setupterm()</code> should be called. Note that <code>setupterm()</code> is automatically called by <code>initscr()</code> and <code>newterm()</code>. This defines the set of terminal-dependent variables (listed in <code>terminfo(4)</code>). The <i>terminfo</i> variables <code>lines</code> and <code>columns</code> are initialized by <code>setupterm()</code> as follows: If <code>use_env(FALSE)</code> has been called, values for <code>lines</code> and <code>columns</code> specified in <i>terminfo</i> are used. Otherwise, if the environment variables <code>LINES</code> and <code>COLUMNS</code> exist, their values are used. If these environment variables do not exist and the program is running in a window, the current window size is used. Otherwise, if the environment variables do not exist, the values for <code>lines</code> and <code>columns</code> specified in the <i>terminfo</i> database are used.</p> <p>The headers <code><curses.h></code> and <code><term.h></code> should be included (in this order) to get the definitions for these strings, numbers, and flags. Parameterized strings should be passed through <code>tparm()</code> to instantiate them. All <i>terminfo</i> strings (including the output</p> |

tigetstr(3CURSES)

of `tparm()` should be printed with `tputs()` or `putp()`. Call the `reset_shell_mode()` routine to restore the tty modes before exiting (see `curs_kernel(3CURSES)`). Programs which use cursor addressing should output `enter_ca_mode` upon startup and should output `exit_ca_mode` before exiting. Programs desiring shell escapes should call `reset_shell_mode` and output `exit_ca_mode` before the shell is called and should output `enter_ca_mode` and call `reset_prog_mode` after returning from the shell.

The `setupterm()` routine reads in the *terminfo* database, initializing the *terminfo* structures, but does not set up the output virtualization structures used by *curses*. The terminal type is the character string *term*; if *term* is null, the environment variable `TERM` is used. All output is to file descriptor *fildev* which is initialized for output. If *errret* is not null, then `setupterm()` returns `OK` or `ERR` and stores a status value in the integer pointed to by *errret*. A status of 1 in *errret* is normal, 0 means that the terminal could not be found, and -1 means that the *terminfo* database could not be found. If *errret* is null, `setupterm()` prints an error message upon finding an error and exits. Thus, the simplest call is:

```
setupterm((char *)0, 1, (int *)0);
```

which uses all the defaults and sends the output to `stdout`.

The `setterm()` routine is being replaced by `setupterm()`. The call:

```
setupterm(term, 1, (int *)0)
```

provides the same functionality as `setterm(term)`. The `setterm()` routine is included here for compatibility and is supported at Level 2.

The `set_curterm()` routine sets the variable `cur_term` to *nterm*, and makes all of the *terminfo* boolean, numeric, and string variables use the values from *nterm*.

The `del_curterm()` routine frees the space pointed to by *oterm* and makes it available for further use. If *oterm* is the same as `cur_term`, references to any of the *terminfo* boolean, numeric, and string variables thereafter may refer to invalid memory locations until another `setupterm()` has been called.

The `restartterm()` routine is similar to `setupterm()` and `initscr()`, except that it is called after restoring memory to a previous state. It assumes that the windows and the input and output options are the same as when memory was saved, but the terminal type and baud rate may be different.

The `tparm()` routine instantiates the string *str* with parameters *pi*. A pointer is returned to the result of *str* with the parameters applied.

The `tputs()` routine applies padding information to the string *str* and outputs it. The *str* must be a *terminfo* string variable or the return value from `tparm()`, `tgetstr()`, or `tgoto()`. *affcnt* is the number of lines affected, or 1 if not applicable. *putc* is a `putchar()`-like routine to which the characters are passed, one at a time.

tigetstr(3CURSES)

The `putp()` routine calls `tputs(str, 1, putchar)`. Note that the output of `putpA()` always goes to `stdout`, not to the *files* specified in `setupterm()`.

The `vidputs()` routine displays the string on the terminal in the video attribute mode *attrs*, which is any combination of the attributes listed in `curses(3CURSES)`. The characters are passed to the `putchar()`-like routine `putc()`.

The `vidattr()` routine is like the `vidputs()` routine, except that it outputs through `putchar()`.

The `mvcur()` routine provides low-level cursor motion.

The `tigetflag()`, `tigetnum()` and `tigetstr()` routines return the value of the capability corresponding to the *terminfo capname* passed to them, such as `xenl`.

With the `tigetflag()` routine, the value `-1` is returned if *capname* is not a boolean capability.

With the `tigetnum()` routine, the value `-2` is returned if *capname* is not a numeric capability.

With the `tigetstr()` routine, the value `(char *)-1` is returned if *capname* is not a string capability.

The *capname* for each capability is given in the table column entitled *capname* code in the capabilities section of `terminfo(4)`.

```
char *boolnames, *boolcodes, *boolfnames
char *numnames, *numcodes, *numfnames
char *strnames, *strcodes, *strfnames
```

These null-terminated arrays contain the *capnames*, the *termcap* codes, and the full C names, for each of the *terminfo* variables.

RETURN VALUES

All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

Routines that return pointers always return `NULL` on error.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_initscr(3CURSES)`, `curs_kernel(3CURSES)`, `curs_termcap(3CURSES)`, `curses(3CURSES)`, `putc(3C)`, `terminfo(4)`, `attributes(5)`

tigetstr(3CURSES)

NOTES | The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

The `setupterm()` routine should be used in place of `setterm()`.

Note that `vidattr()` and `vidputs()` may be macros.

| | |
|----------------------|---|
| NAME | tigetflag, tigetnum, tigetstr, tparm – return the value of a terminfo capability |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <term.h> int tigetflag(char *capname); int tigetnum(char *capname); char *tigetstr(char *capname); char *tparm(char *cap, long p1, long p2, long p3, long p4, long p5, long p6, long p7, long p8, long p9);</pre> |
| PARAMETERS | <p><i>capname</i> Is the name of the terminfo capability for which the value is required.</p> <p><i>cap</i> Is a pointer to a string capability.</p> <p><i>p1...p9</i> Are the parameters to be instantiated.</p> |
| DESCRIPTION | <p>The tigetflag(), tigetnum(), and tigetstr() functions return values for terminfo capabilities passed to them.</p> <p>The following null-terminated arrays contain the <i>capnames</i>, the termcap codes and full C names for each of the terminfo variables.</p> <pre>char *boolnames, *boolcodes, *boolfnames char *numnames, *numcodes, *numfnames char *strnames, *strcodes, *strfnames</pre> <p>The tparm() function instantiates a parameterized string using nine arguments. The string is suitable for output processing by tputs().</p> |
| RETURN VALUES | <p>On success, the tigetflg(), tigetnum(), and tigetstr() functions return the specified terminfo capability.</p> <p>tigetflag() returns -1 if <i>capname</i> is not a Boolean capability.</p> <p>tigetnum() returns -2 if <i>capname</i> is not a numeric capability.</p> <p>tigetstr() returns (char *)-1 if <i>capname</i> is not a string capability.</p> <p>On success, the tparm() function returns <i>cap</i> in a static buffer with the parameterization resolved. Otherwise, it returns a null pointer.</p> |
| ERRORS | None. |

tigetstr(3XCURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `libcurses(3XCURSES)`, `tgetent(3XCURSES)`, `terminfo(4)`, `attributes(5)`, `standards(5)`

timeout(3CURSES)

| | |
|--------------------|--|
| NAME | <code>curs_inopts, cbreak, nocbreak, echo, noecho, halfdelay, intrflush, keypad, meta, nodelay, notimeout, raw, noraw, noqiflush, qiflush, timeout, wtimeout, typeahead – curses terminal input option control routines</code> |
| SYNOPSIS | <pre>cc [<i>flag ...</i>] <i>file ...</i> -lcurses [<i>library ...</i>] #include <curses.h> int cbreak(void); int nocbreak(void); int echo(void); int noecho(void); int halfdelay(int <i>tenths</i>); int intrflush(WINDOW *<i>win</i>, bool <i>bf</i>); int keypad(WINDOW *<i>win</i>, bool <i>bf</i>); int meta(WINDOW *<i>win</i>, bool <i>bf</i>); int nodelay(WINDOW *<i>win</i>, bool <i>bf</i>); int notimeout(WINDOW *<i>win</i>, bool <i>bf</i>); int raw(void); int noraw(void); void noqiflush(void); void qiflush(void); void timeout(int <i>delay</i>); void wtimeout(WINDOW *<i>win</i>, int <i>delay</i>); int typeahead(int <i>fildev</i>);</pre> |
| DESCRIPTION | <p>The <code>cbreak()</code> and <code>nocbreak()</code> routines put the terminal into and out of <code>cbreak()</code> mode, respectively. In this mode, characters typed by the user are immediately available to the program, and erase/kill character-processing is not performed. When out of this mode, the tty driver buffers the typed characters until a newline or carriage return is typed. Interrupt and flow control characters are unaffected by this mode. Initially the terminal may or may not be in <code>cbreak()</code> mode, as the mode is inherited; therefore, a program should call <code>cbreak()</code> or <code>nocbreak()</code> explicitly. Most interactive programs using <code>curses</code> set the <code>cbreak()</code> mode.</p> <p>Note that <code>cbreak()</code> overrides <code>raw()</code>. (See <code>curs_getch(3CURSES)</code> for a discussion of how these routines interact with <code>echo()</code> and <code>noecho()</code>.)</p> <p>The <code>echo()</code> and <code>noecho()</code> routines control whether characters typed by the user are echoed by <code>getch()</code> as they are typed. Echoing by the tty driver is always disabled, but initially <code>getch()</code> is in echo mode, so characters typed are echoed. Authors of</p> |

timeout(3CURSES)

most interactive programs prefer to do their own echoing in a controlled area of the screen, or not to echo at all, so they disable echoing by calling `noecho()`. (See `curs_getch(3CURSES)` for a discussion of how these routines interact with `cbreak()` and `nocbreak()`.)

The `halfdelay()` routine is used for half-delay mode, which is similar to `cbreak()` mode in that characters typed by the user are immediately available to the program. However, after blocking for *tenths* tenths of seconds, `ERR` is returned if nothing has been typed. The value of *tenths* must be a number between 1 and 255. Use `nocbreak()` to leave half-delay mode.

If the `intrflush()` option is enabled, (*bf* is `TRUE`), when an interrupt key is pressed on the keyboard (interrupt, break, quit) all output in the tty driver queue will be flushed, giving the effect of faster response to the interrupt, but causing `curses` to have the wrong idea of what is on the screen. Disabling (*bf* is `FALSE`), the option prevents the flush. The default for the option is inherited from the tty driver settings. The window argument is ignored.

The `keypad()` option enables the keypad of the user's terminal. If enabled (*bf* is `TRUE`), the user can press a function key (such as an arrow key) and `wgetch()` returns a single value representing the function key, as in `KEY_LEFT`. If disabled (*bf* is `FALSE`), `curses` does not treat function keys specially and the program has to interpret the escape sequences itself. If the keypad in the terminal can be turned on (made to transmit) and off (made to work locally), turning on this option causes the terminal keypad to be turned on when `wgetch()` is called. The default value for keypad is false.

Initially, whether the terminal returns 7 or 8 significant bits on input depends on the control mode of the tty driver (see `termio(7I)`). To force 8 bits to be returned, invoke `meta(win, TRUE)`. To force 7 bits to be returned, invoke `meta(win, FALSE)`. The window argument, *win*, is always ignored. If the terminfo capabilities `smm` (`meta_on`) and `rmm` (`meta_off`) are defined for the terminal, `smm` is sent to the terminal when `meta(win, TRUE)` is called and `rmm` is sent when `meta(win, FALSE)` is called.

The `nodelay()` option causes `getch()` to be a non-blocking call. If no input is ready, `getch()` returns `ERR`. If disabled (*bf* is `FALSE`), `getch()` waits until a key is pressed.

While interpreting an input escape sequence, `wgetch()` sets a timer while waiting for the next character. If `notimeout(win, TRUE)` is called, then `wgetch()` does not set a timer. The purpose of the timeout is to differentiate between sequences received from a function key and those typed by a user.

With the `raw()` and `noraw()` routines, the terminal is placed into or out of raw mode. Raw mode is similar to `cbreak()` mode, in that characters typed are immediately passed through to the user program. The differences are that in raw mode, the interrupt, quit, suspend, and flow control characters are all passed through uninterpreted, instead of generating a signal. The behavior of the `BREAK` key depends on other bits in the tty driver that are not set by `curses`.

timeout(3CURSES)

When the `noqiflush()` routine is used, normal flush of input and output queues associated with the `INTR`, `QUIT` and `SUSP` characters will not be done (see `termio(7I)`). When `qiflush()` is called, the queues will be flushed when these control characters are read.

The `timeout()` and `wtimeout()` routines set blocking or non-blocking read for a given window. If *delay* is negative, blocking read is used (that is, waits indefinitely for input). If *delay* is zero, then non-blocking read is used (that is, read returns `ERR` if no input is waiting). If *delay* is positive, then read blocks for *delay* milliseconds, and returns `ERR` if there is still no input. Hence, these routines provide the same functionality as `nodelay()`, plus the additional capability of being able to block for only *delay* milliseconds (where *delay* is positive).

`curses` does “line-breakout optimization” by looking for typeahead periodically while updating the screen. If input is found, and it is coming from a `tty`, the current update is postponed until `refresh()` or `doupdate()` is called again. This allows faster response to commands typed in advance. Normally, the input `FILE` pointer passed to `newterm()`, or `stdin` in the case that `initscr()` was used, will be used to do this typeahead checking. The `typeahead()` routine specifies that the file descriptor *fildev* is to be used to check for typeahead instead. If *fildev* is `-1`, then no typeahead checking is done.

RETURN VALUES

All routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_getch(3CURSES)`, `curs_initscr(3CURSES)`, `curses(3CURSES)`, `attributes(5)`, `termio(7I)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `echo()`, `noecho()`, `halfdelay()`, `intrflush()`, `meta()`, `nodelay()`, `notimeout()`, `noqiflush()`, `qiflush()`, `timeout()`, and `wtimeout()` may be macros.

timeout(3XCURSES)

NAME notimeout, timeout, wtimeout – set timed blocking or non-blocking read

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

int notimeout(WINDOW *win, bool bf);

void timeout(int delay);

void wtimeout(WINDOW win, int delay);
```

PARAMETERS

| | |
|--------------|--|
| <i>win</i> | Is a pointer to the window in which to set the timed blocking. |
| <i>bf</i> | Is a Boolean expression. |
| <i>delay</i> | Is the number of milliseconds to block or wait for input. |

DESCRIPTION

If *bool* is TRUE, the notimeout() function disables a timer used by getch(3XCURSES) when handling multibyte function key sequences.

When *bool* is FALSE and keypad handling is enabled, a timer is set by getch() to handle bytes received that could be the beginning of a function key (for example, ESC). If the remainder of the sequence is not received before the time expires, the first byte is returned; otherwise, the value of the function key is returned. Subsequent calls to the getch() function will return the other bytes received for the incomplete key sequence.

The timeout() and wtimeout() functions set the length of time getch() waits for input for windows stdscr and win, respectively. These functions are similar to nodelay(3XCURSES) except the time to block or wait for input can be specified.

A negative *delay* causes the program to wait indefinitely for input; a *delay* of 0 returns ERR if no input is ready; and a positive *delay* blocks until input arrives or the time specified expires, (in which case, ERR is returned).

RETURN VALUES On success, the notimeout() function returns OK. Otherwise, it returns ERR.

The timeout() and wtimeout() functions do not return a value.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |

timeout(3XCURSES)

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO getch(3XCURSES), halfdelay(3XCURSES), libcurses(3XCURSES),
nodelay(3XCURSES), attributes(5), standards(5)

top_panel(3CURSES)

NAME panel_top, top_panel, bottom_panel – panels deck manipulation routines

SYNOPSIS

```
cc [ flag ... ] file ... -lpanel -lcurses [ library .. ]
#include <panel.h>

int top_panel (PANEL *panel) ;
int bottom_panel (PANEL *panel) ;
```

DESCRIPTION top_panel () pulls *panel* to the top of the desk of panels. It leaves the size, location, and contents of its associated window unchanged.

bottom_panel () puts *panel* at the bottom of the deck of panels. It leaves the size, location, and contents of its associated window unchanged.

RETURN VALUES All of these routines return the integer OK upon successful completion or ERR upon error.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), panel_update(3CURSES), panels(3CURSES), attributes(5)

NOTES The header <panel.h> automatically includes the header <curses.h>.

top_row(3CURSES)

NAME menu_item_current, set_current_item, current_item, set_top_row, top_row, item_index
– set and get current menus items

SYNOPSIS

```
cc [ flag... ] file... -lmenu -lcurses [ library... ]
#include <menu.h>

int set_current_item(MENU *menu, ITEM *item);
ITEM *current_item(MENU *menu);
int set_top_row(MENU *menu, int row);
int top_row(MENU *menu);
int item_index(ITEM *item);
```

DESCRIPTION

The current item of a menu is the item where the cursor is currently positioned. `set_current_item()` sets the current item of *menu* to *item*. `current_item()` returns a pointer to the the current item in *menu*.

`set_top_row()` sets the top row of *menu* to *row*. The left-most item on the new top row becomes the current item. `top_row()` returns the number of the menu row currently displayed at the top of *menu*.

`item_index()` returns the index to the *item* in the item pointer array. The value of this index ranges from 0 through *N*-1, where *N* is the total number of items connected to the menu.

RETURN VALUES

`current_item()` returns NULL on error.

`top_row()` and `index_item()` return -1 on error.

`set_current_item()` and `set_top_row()` return one of the following:

| | |
|-----------------|--|
| E_OK | The routine returned successfully. |
| E_SYSTEM_ERROR | System error. |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. |
| E_BAD_STATE | The routine was called from an initialization or termination function. |
| E_NOT_CONNECTED | No items are connected to the menu. |

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3CURSES)`, `menus(3CURSES)`, `attributes(5)`

top_row(3CURSES)

NOTES | The header `<menu.h>` automatically includes the headers `<eti.h>` and `<curses.h>`.

| NAME | curs_touch, touchwin, touchline, untouchwin, wtouchln, is_linetouched, is_wintouched – curses refresh control routines | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int touchwin(WINDOW *win); int touchline(WINDOW *win, int start, int count); int untouchwin(WINDOW *win); int wtouchln(WINDOW *win, int y, int n, int changed); int is_linetouched(WINDOW *win, int line); int is_wintouched(WINDOW *win);</pre> | | | | |
| DESCRIPTION | <p>The touchwin() and touchline() routines throw away all optimization information about which parts of the window have been touched, by pretending that the entire window has been drawn on. This is sometimes necessary when using overlapping windows, since a change to one window affects the other window, but the records of which lines have been changed in the other window do not reflect the change. The routine touchline() only pretends that <i>count</i> lines have been changed, beginning with line <i>start</i>.</p> <p>The untouchwin() routine marks all lines in the window as unchanged since the last call to wrefresh().</p> <p>The wtouchln() routine makes <i>n</i> lines in the window, starting at line <i>y</i>, look as if they have (<i>changed</i>=1) or have not (<i>changed</i>=0) been changed since the last call to wrefresh().</p> <p>The is_linetouched() and is_wintouched() routines return TRUE if the specified line/window was modified since the last call to wrefresh(); otherwise they return FALSE. In addition, is_linetouched() returns ERR if line is not valid for the given window.</p> | | | | |
| RETURN VALUES | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion, unless otherwise noted in the preceding routine descriptions. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curs_refresh(3CURSES), curses(3CURSES), attributes(5) | | | | |
| NOTES | The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>. | | | | |

`touchline(3CURSES)`

Note that all routines except `wtouchln()` may be macros.

| | | | | | | | | | | | | | | | |
|--------------------|--|------------|---|-------------|---|--------------|--|--------------|--|----------|---|----------|--|----------------|---|
| NAME | is_linetouched, is_wintouched, touchline, touchwin, untouchwin, wtouchln – control window refresh | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> bool is_linetouched(WINDOW *win, int line); bool is_wintouched(WINDOW *win); int touchline(WINDOW *win, int start, int count); int touchwin(WINDOW *win); int untouchwin(WINDOW *win); int wtouchln(WINDOW *win, int y, int n, int changed);</pre> | | | | | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>win</i></td> <td>Is a pointer to the window in which the refresh is to be controlled or monitored.</td> </tr> <tr> <td><i>line</i></td> <td>Is the line to be checked for change since refresh.</td> </tr> <tr> <td><i>start</i></td> <td>Is the starting line number of the portion of the window to make appear changed.</td> </tr> <tr> <td><i>count</i></td> <td>Is the number of lines in the window to mark as changed.</td> </tr> <tr> <td><i>y</i></td> <td>Is the starting line number of the portion of the window to make appear changed or not changed.</td> </tr> <tr> <td><i>n</i></td> <td>Is the number of lines in the window to mark as changed.</td> </tr> <tr> <td><i>changed</i></td> <td>Is a flag indicating whether to make lines look changed (0) or not changed (1).</td> </tr> </table> | <i>win</i> | Is a pointer to the window in which the refresh is to be controlled or monitored. | <i>line</i> | Is the line to be checked for change since refresh. | <i>start</i> | Is the starting line number of the portion of the window to make appear changed. | <i>count</i> | Is the number of lines in the window to mark as changed. | <i>y</i> | Is the starting line number of the portion of the window to make appear changed or not changed. | <i>n</i> | Is the number of lines in the window to mark as changed. | <i>changed</i> | Is a flag indicating whether to make lines look changed (0) or not changed (1). |
| <i>win</i> | Is a pointer to the window in which the refresh is to be controlled or monitored. | | | | | | | | | | | | | | |
| <i>line</i> | Is the line to be checked for change since refresh. | | | | | | | | | | | | | | |
| <i>start</i> | Is the starting line number of the portion of the window to make appear changed. | | | | | | | | | | | | | | |
| <i>count</i> | Is the number of lines in the window to mark as changed. | | | | | | | | | | | | | | |
| <i>y</i> | Is the starting line number of the portion of the window to make appear changed or not changed. | | | | | | | | | | | | | | |
| <i>n</i> | Is the number of lines in the window to mark as changed. | | | | | | | | | | | | | | |
| <i>changed</i> | Is a flag indicating whether to make lines look changed (0) or not changed (1). | | | | | | | | | | | | | | |
| DESCRIPTION | <p>The touchwin() function marks the entire window as dirty. This makes it appear to X/Open Curses as if the whole window has been changed, thus causing the entire window to be rewritten with the next call to refresh(3XCURSES). This is sometimes necessary when using overlapping windows; the change to one window will not be reflected in the other and, hence will not be recorded.</p> <p>The touchline() function marks as dirty a portion of the window starting at line <i>start</i> and continuing for <i>count</i> lines instead of the entire window. Consequently, that portion of the window is updated with the next call to refresh().</p> <p>The untouchwin() function marks all lines in the window as unchanged since the last refresh, ensuring that it is not updated.</p> | | | | | | | | | | | | | | |

touchline(3XCURSES)

The `wtouchln()` function marks *n* lines starting at line *y* as either changed (*changed=1*) or unchanged (*changed=0*) since the last refresh.

To find out which lines or windows have been changed since the last refresh, use the `is_linetouched()` and `is_wintouched()` functions, respectively. These return `TRUE` if the specified line or window have been changed since the last call to `refresh()` or `FALSE` if no changes have been made.

RETURN VALUES On success, these functions return `OK`. Otherwise, they return `ERR`.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `douupdate(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_touch, touchwin, touchline, untouchwin, wtouchln, is_linetouched, is_wintouched – curses refresh control routines | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int touchwin(WINDOW *win); int touchline(WINDOW *win, int start, int count); int untouchwin(WINDOW *win); int wtouchln(WINDOW *win, int y, int n, int changed); int is_linetouched(WINDOW *win, int line); int is_wintouched(WINDOW *win);</pre> | | | | |
| DESCRIPTION | <p>The touchwin() and touchline() routines throw away all optimization information about which parts of the window have been touched, by pretending that the entire window has been drawn on. This is sometimes necessary when using overlapping windows, since a change to one window affects the other window, but the records of which lines have been changed in the other window do not reflect the change. The routine touchline() only pretends that <i>count</i> lines have been changed, beginning with line <i>start</i>.</p> <p>The untouchwin() routine marks all lines in the window as unchanged since the last call to wrefresh().</p> <p>The wtouchln() routine makes <i>n</i> lines in the window, starting at line <i>y</i>, look as if they have (<i>changed</i>=1) or have not (<i>changed</i>=0) been changed since the last call to wrefresh().</p> <p>The is_linetouched() and is_wintouched() routines return TRUE if the specified line/window was modified since the last call to wrefresh(); otherwise they return FALSE. In addition, is_linetouched() returns ERR if line is not valid for the given window.</p> | | | | |
| RETURN VALUES | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion, unless otherwise noted in the preceding routine descriptions. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curs_refresh(3CURSES), curses(3CURSES), attributes(5) | | | | |
| NOTES | The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>. | | | | |

touchwin(3CURSES)

Note that all routines except `wtouchln()` may be macros.

| | | | | | | | | | | | | | | | |
|--------------------|--|------------|---|-------------|---|--------------|--|--------------|--|----------|---|----------|--|----------------|---|
| NAME | <code>is_linetouched</code> , <code>is_wintouched</code> , <code>touchline</code> , <code>touchwin</code> , <code>untouchwin</code> , <code>wtouchln</code> – control window refresh | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> bool is_linetouched(WINDOW *win, int line); bool is_wintouched(WINDOW *win); int touchline(WINDOW *win, int start, int count); int touchwin(WINDOW *win); int untouchwin(WINDOW *win); int wtouchln(WINDOW *win, int y, int n, int changed);</pre> | | | | | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>win</i></td> <td>Is a pointer to the window in which the refresh is to be controlled or monitored.</td> </tr> <tr> <td><i>line</i></td> <td>Is the line to be checked for change since refresh.</td> </tr> <tr> <td><i>start</i></td> <td>Is the starting line number of the portion of the window to make appear changed.</td> </tr> <tr> <td><i>count</i></td> <td>Is the number of lines in the window to mark as changed.</td> </tr> <tr> <td><i>y</i></td> <td>Is the starting line number of the portion of the window to make appear changed or not changed.</td> </tr> <tr> <td><i>n</i></td> <td>Is the number of lines in the window to mark as changed.</td> </tr> <tr> <td><i>changed</i></td> <td>Is a flag indicating whether to make lines look changed (0) or not changed (1).</td> </tr> </table> | <i>win</i> | Is a pointer to the window in which the refresh is to be controlled or monitored. | <i>line</i> | Is the line to be checked for change since refresh. | <i>start</i> | Is the starting line number of the portion of the window to make appear changed. | <i>count</i> | Is the number of lines in the window to mark as changed. | <i>y</i> | Is the starting line number of the portion of the window to make appear changed or not changed. | <i>n</i> | Is the number of lines in the window to mark as changed. | <i>changed</i> | Is a flag indicating whether to make lines look changed (0) or not changed (1). |
| <i>win</i> | Is a pointer to the window in which the refresh is to be controlled or monitored. | | | | | | | | | | | | | | |
| <i>line</i> | Is the line to be checked for change since refresh. | | | | | | | | | | | | | | |
| <i>start</i> | Is the starting line number of the portion of the window to make appear changed. | | | | | | | | | | | | | | |
| <i>count</i> | Is the number of lines in the window to mark as changed. | | | | | | | | | | | | | | |
| <i>y</i> | Is the starting line number of the portion of the window to make appear changed or not changed. | | | | | | | | | | | | | | |
| <i>n</i> | Is the number of lines in the window to mark as changed. | | | | | | | | | | | | | | |
| <i>changed</i> | Is a flag indicating whether to make lines look changed (0) or not changed (1). | | | | | | | | | | | | | | |
| DESCRIPTION | <p>The <code>touchwin()</code> function marks the entire window as dirty. This makes it appear to X/Open Curses as if the whole window has been changed, thus causing the entire window to be rewritten with the next call to <code>refresh(3XCURSES)</code>. This is sometimes necessary when using overlapping windows; the change to one window will not be reflected in the other and, hence will not be recorded.</p> <p>The <code>touchline()</code> function marks as dirty a portion of the window starting at line <i>start</i> and continuing for <i>count</i> lines instead of the entire window. Consequently, that portion of the window is updated with the next call to <code>refresh()</code>.</p> <p>The <code>untouchwin()</code> function marks all lines in the window as unchanged since the last refresh, ensuring that it is not updated.</p> | | | | | | | | | | | | | | |

touchwin(3XCURSES)

The `wtouchln()` function marks *n* lines starting at line *y* as either changed (*changed*=1) or unchanged (*changed*=0) since the last refresh.

To find out which lines or windows have been changed since the last refresh, use the `is_linetouched()` and `is_wintouched()` functions, respectively. These return `TRUE` if the specified line or window have been changed since the last call to `refresh()` or `FALSE` if no changes have been made.

RETURN VALUES On success, these functions return `OK`. Otherwise, they return `ERR`.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `douupdate(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

tparm(3CURSES)

| | |
|-------------|--|
| NAME | curs_terminfo, setupterm, setterm, set_curterm, del_curterm, restartterm, tparm, tputs, putp, vidputs, vidattr, mvcur, tigetflag, tigetnum, tigetstr – curses interfaces to terminfo database |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> #include <term.h> int setupterm(char *term, int fildes, int *errret); int setterm(char *term); int set_curterm(TERMINAL *nterm); int del_curterm(TERMINAL *oterm); int restartterm(char *term, int fildes, int *errret); char *tparm(char *str, long int p1, long int p2, long int p3, long int p4, long int p5, long int p6, long int p7, long int p8, long int p9); int tputs(char *str, int affcnt, int (*putc)(char)); int putp(char *str); int vidputs(chtype attrs, int (*putc)(char)); int vidattr(chtype attrs); int mvcur(int oldrow, int oldcol, int newrow, int newcol); int tigetflag(char *capname); int tigetnum(char *capname); char *tigetstr(char *capname);</pre> |
| DESCRIPTION | <p>These low-level routines must be called by programs that have to deal directly with the <i>terminfo</i> database to handle certain terminal capabilities, such as programming function keys. For all other functionality, curses routines are more suitable and their use is recommended.</p> <p>Initially, <code>setupterm()</code> should be called. Note that <code>setupterm()</code> is automatically called by <code>initscr()</code> and <code>newterm()</code>. This defines the set of terminal-dependent variables (listed in <code>terminfo(4)</code>). The <i>terminfo</i> variables <code>lines</code> and <code>columns</code> are initialized by <code>setupterm()</code> as follows: If <code>use_env(FALSE)</code> has been called, values for <code>lines</code> and <code>columns</code> specified in <i>terminfo</i> are used. Otherwise, if the environment variables <code>LINES</code> and <code>COLUMNS</code> exist, their values are used. If these environment variables do not exist and the program is running in a window, the current window size is used. Otherwise, if the environment variables do not exist, the values for <code>lines</code> and <code>columns</code> specified in the <i>terminfo</i> database are used.</p> <p>The headers <code><curses.h></code> and <code><term.h></code> should be included (in this order) to get the definitions for these strings, numbers, and flags. Parameterized strings should be passed through <code>tparm()</code> to instantiate them. All <i>terminfo</i> strings (including the output</p> |

tparm(3CURSES)

of `tparm()` should be printed with `tputs()` or `putp()`. Call the `reset_shell_mode()` routine to restore the tty modes before exiting (see `curls_kernel(3CURSES)`). Programs which use cursor addressing should output `enter_ca_mode` upon startup and should output `exit_ca_mode` before exiting. Programs desiring shell escapes should call `reset_shell_mode` and output `exit_ca_mode` before the shell is called and should output `enter_ca_mode` and call `reset_prog_mode` after returning from the shell.

The `setupterm()` routine reads in the *terminfo* database, initializing the *terminfo* structures, but does not set up the output virtualization structures used by *curses*. The terminal type is the character string *term*; if *term* is null, the environment variable `TERM` is used. All output is to file descriptor *fildev* which is initialized for output. If *errret* is not null, then `setupterm()` returns `OK` or `ERR` and stores a status value in the integer pointed to by *errret*. A status of 1 in *errret* is normal, 0 means that the terminal could not be found, and -1 means that the *terminfo* database could not be found. If *errret* is null, `setupterm()` prints an error message upon finding an error and exits. Thus, the simplest call is:

```
setupterm((char *)0, 1, (int *)0);
```

which uses all the defaults and sends the output to `stdout`.

The `setterm()` routine is being replaced by `setupterm()`. The call:

```
setupterm(term, 1, (int *)0)
```

provides the same functionality as `setterm(term)`. The `setterm()` routine is included here for compatibility and is supported at Level 2.

The `set_curterm()` routine sets the variable `cur_term` to *nterm*, and makes all of the *terminfo* boolean, numeric, and string variables use the values from *nterm*.

The `del_curterm()` routine frees the space pointed to by *oterm* and makes it available for further use. If *oterm* is the same as `cur_term`, references to any of the *terminfo* boolean, numeric, and string variables thereafter may refer to invalid memory locations until another `setupterm()` has been called.

The `restartterm()` routine is similar to `setupterm()` and `initscr()`, except that it is called after restoring memory to a previous state. It assumes that the windows and the input and output options are the same as when memory was saved, but the terminal type and baud rate may be different.

The `tparm()` routine instantiates the string *str* with parameters *pi*. A pointer is returned to the result of *str* with the parameters applied.

The `tputs()` routine applies padding information to the string *str* and outputs it. The *str* must be a *terminfo* string variable or the return value from `tparm()`, `tgetstr()`, or `tgoto()`. *affcnt* is the number of lines affected, or 1 if not applicable. *putc* is a `putchar()`-like routine to which the characters are passed, one at a time.

tparm(3CURSES)

The `putp()` routine calls `tputs(str, 1, putchar)`. Note that the output of `putpA()` always goes to `stdout`, not to the *files* specified in `setupterm()`.

The `vidputs()` routine displays the string on the terminal in the video attribute mode *attrs*, which is any combination of the attributes listed in `curses(3CURSES)`. The characters are passed to the `putchar()`-like routine `putc()`.

The `vidattr()` routine is like the `vidputs()` routine, except that it outputs through `putchar()`.

The `mvcur()` routine provides low-level cursor motion.

The `tigetflag()`, `tigetnum()` and `tigetstr()` routines return the value of the capability corresponding to the *terminfo capname* passed to them, such as `xenl`.

With the `tigetflag()` routine, the value `-1` is returned if *capname* is not a boolean capability.

With the `tigetnum()` routine, the value `-2` is returned if *capname* is not a numeric capability.

With the `tigetstr()` routine, the value `(char *)-1` is returned if *capname* is not a string capability.

The *capname* for each capability is given in the table column entitled *capname* code in the capabilities section of `terminfo(4)`.

```
char *boolnames, *boolcodes, *boolfnames
char *numnames, *numcodes, *numfnames
char *strnames, *strcodes, *strfnames
```

These null-terminated arrays contain the *capnames*, the *termcap* codes, and the full C names, for each of the *terminfo* variables.

RETURN VALUES

All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

Routines that return pointers always return `NULL` on error.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_initscr(3CURSES)`, `curs_kernel(3CURSES)`, `curs_termcap(3CURSES)`, `curses(3CURSES)`, `putc(3C)`, `terminfo(4)`, `attributes(5)`

`tparm(3CURSES)`

NOTES | The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

The `setupterm()` routine should be used in place of `setterm()`.

Note that `vidattr()` and `vidputs()` may be macros.

| | |
|----------------------|---|
| NAME | tigetflag, tigetnum, tigetstr, tparm – return the value of a terminfo capability |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <term.h> int tigetflag(char *capname); int tigetnum(char *capname); char *tigetstr(char *capname); char *tparm(char *cap, long p1, long p2, long p3, long p4, long p5, long p6, long p7, long p8, long p9);</pre> |
| PARAMETERS | <p><i>capname</i> Is the name of the terminfo capability for which the value is required.</p> <p><i>cap</i> Is a pointer to a string capability.</p> <p><i>p1...p9</i> Are the parameters to be instantiated.</p> |
| DESCRIPTION | <p>The <code>tigetflag()</code>, <code>tigetnum()</code>, and <code>tigetstr()</code> functions return values for terminfo capabilities passed to them.</p> <p>The following null-terminated arrays contain the <i>capnames</i>, the termcap codes and full C names for each of the terminfo variables.</p> <pre>char *boolnames, *boolcodes, *boolfnames char *numnames, *numcodes, *numfnames char *strnames, *strcodes, *strfnames</pre> <p>The <code>tparm()</code> function instantiates a parameterized string using nine arguments. The string is suitable for output processing by <code>tputs()</code>.</p> |
| RETURN VALUES | <p>On success, the <code>tigetflg()</code>, <code>tigetnum()</code>, and <code>tigetstr()</code> functions return the specified terminfo capability.</p> <p><code>tigetflag()</code> returns <code>-1</code> if <i>capname</i> is not a Boolean capability.</p> <p><code>tigetnum()</code> returns <code>-2</code> if <i>capname</i> is not a numeric capability.</p> <p><code>tigetstr()</code> returns <code>(char *)-1</code> if <i>capname</i> is not a string capability.</p> <p>On success, the <code>tparm()</code> function returns <i>cap</i> in a static buffer with the parameterization resolved. Otherwise, it returns a null pointer.</p> |
| ERRORS | None. |

`tparm(3XCURSES)`

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `libcurses(3XCURSES)`, `tgetent(3XCURSES)`, `terminfo(4)`, `attributes(5)`, `standards(5)`

| | |
|--------------------|--|
| NAME | curs_terminfo, setupterm, setterm, set_curterm, del_curterm, restartterm, tparm, tputs, putp, vidputs, vidattr, mvcur, tigetflag, tigetnum, tigetstr – curses interfaces to terminfo database |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> #include <term.h> int setupterm(char *term, int fildes, int *errret); int setterm(char *term); int set_curterm(TERMINAL *nterm); int del_curterm(TERMINAL *oterm); int restartterm(char *term, int fildes, int *errret); char *tparm(char *str, long int p1, long int p2, long int p3, long int p4, long int p5, long int p6, long int p7, long int p8, long int p9); int tputs(char *str, int affcnt, int (*putc)(char)); int putp(char *str); int vidputs(chtype attrs, int (*putc)(char)); int vidattr(chtype attrs); int mvcur(int oldrow, int oldcol, int newrow, int newcol); int tigetflag(char *capname); int tigetnum(char *capname); char *tigetstr(char *capname);</pre> |
| DESCRIPTION | <p>These low-level routines must be called by programs that have to deal directly with the <i>terminfo</i> database to handle certain terminal capabilities, such as programming function keys. For all other functionality, curses routines are more suitable and their use is recommended.</p> <p>Initially, <code>setupterm()</code> should be called. Note that <code>setupterm()</code> is automatically called by <code>initscr()</code> and <code>newterm()</code>. This defines the set of terminal-dependent variables (listed in <code>terminfo(4)</code>). The <i>terminfo</i> variables <code>lines</code> and <code>columns</code> are initialized by <code>setupterm()</code> as follows: If <code>use_env(FALSE)</code> has been called, values for <code>lines</code> and <code>columns</code> specified in <i>terminfo</i> are used. Otherwise, if the environment variables <code>LINES</code> and <code>COLUMNS</code> exist, their values are used. If these environment variables do not exist and the program is running in a window, the current window size is used. Otherwise, if the environment variables do not exist, the values for <code>lines</code> and <code>columns</code> specified in the <i>terminfo</i> database are used.</p> <p>The headers <code><curses.h></code> and <code><term.h></code> should be included (in this order) to get the definitions for these strings, numbers, and flags. Parameterized strings should be passed through <code>tparm()</code> to instantiate them. All <i>terminfo</i> strings (including the output</p> |

tputs(3CURSES)

of `tparm()` should be printed with `tputs()` or `putp()`. Call the `reset_shell_mode()` routine to restore the tty modes before exiting (see `curls_kernel(3CURSES)`). Programs which use cursor addressing should output `enter_ca_mode` upon startup and should output `exit_ca_mode` before exiting. Programs desiring shell escapes should call `reset_shell_mode` and output `exit_ca_mode` before the shell is called and should output `enter_ca_mode` and call `reset_prog_mode` after returning from the shell.

The `setupterm()` routine reads in the *terminfo* database, initializing the *terminfo* structures, but does not set up the output virtualization structures used by *curses*. The terminal type is the character string *term*; if *term* is null, the environment variable `TERM` is used. All output is to file descriptor *fildev* which is initialized for output. If *errret* is not null, then `setupterm()` returns `OK` or `ERR` and stores a status value in the integer pointed to by *errret*. A status of 1 in *errret* is normal, 0 means that the terminal could not be found, and -1 means that the *terminfo* database could not be found. If *errret* is null, `setupterm()` prints an error message upon finding an error and exits. Thus, the simplest call is:

```
setupterm((char *)0, 1, (int *)0);
```

which uses all the defaults and sends the output to `stdout`.

The `setterm()` routine is being replaced by `setupterm()`. The call:

```
setupterm(term, 1, (int *)0)
```

provides the same functionality as `setterm(term)`. The `setterm()` routine is included here for compatibility and is supported at Level 2.

The `set_curterm()` routine sets the variable `cur_term` to *nterm*, and makes all of the *terminfo* boolean, numeric, and string variables use the values from *nterm*.

The `del_curterm()` routine frees the space pointed to by *oterm* and makes it available for further use. If *oterm* is the same as `cur_term`, references to any of the *terminfo* boolean, numeric, and string variables thereafter may refer to invalid memory locations until another `setupterm()` has been called.

The `restartterm()` routine is similar to `setupterm()` and `initscr()`, except that it is called after restoring memory to a previous state. It assumes that the windows and the input and output options are the same as when memory was saved, but the terminal type and baud rate may be different.

The `tparm()` routine instantiates the string *str* with parameters *pi*. A pointer is returned to the result of *str* with the parameters applied.

The `tputs()` routine applies padding information to the string *str* and outputs it. The *str* must be a *terminfo* string variable or the return value from `tparm()`, `tgetstr()`, or `tgoto()`. *affcnt* is the number of lines affected, or 1 if not applicable. *putc* is a `putchar()`-like routine to which the characters are passed, one at a time.

tputs(3CURSES)

The `putp()` routine calls `tputs(str, 1, putchar)`. Note that the output of `putpA()` always goes to `stdout`, not to the *files* specified in `setupterm()`.

The `vidputs()` routine displays the string on the terminal in the video attribute mode *attrs*, which is any combination of the attributes listed in `curses(3CURSES)`. The characters are passed to the `putchar()`-like routine `putc()`.

The `vidattr()` routine is like the `vidputs()` routine, except that it outputs through `putchar()`.

The `mvcur()` routine provides low-level cursor motion.

The `tigetflag()`, `tigetnum()` and `tigetstr()` routines return the value of the capability corresponding to the *terminfo capname* passed to them, such as `xenl`.

With the `tigetflag()` routine, the value `-1` is returned if *capname* is not a boolean capability.

With the `tigetnum()` routine, the value `-2` is returned if *capname* is not a numeric capability.

With the `tigetstr()` routine, the value `(char *)-1` is returned if *capname* is not a string capability.

The *capname* for each capability is given in the table column entitled *capname* code in the capabilities section of `terminfo(4)`.

```
char *boolnames, *boolcodes, *boolfnames
char *numnames, *numcodes, *numfnames
char *strnames, *strcodes, *strfnames
```

These null-terminated arrays contain the *capnames*, the *termcap* codes, and the full C names, for each of the *terminfo* variables.

RETURN VALUES

All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

Routines that return pointers always return `NULL` on error.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_initscr(3CURSES)`, `curs_kernel(3CURSES)`, `curs_termcap(3CURSES)`, `curses(3CURSES)`, `putc(3C)`, `terminfo(4)`, `attributes(5)`

tputs(3CURSES)

NOTES | The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

The `setupterm()` routine should be used in place of `setterm()`.

Note that `vidattr()` and `vidputs()` may be macros.

| | |
|----------------------|--|
| NAME | putp, tputs – apply padding information and output string |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int putp(const char *str); int tputs(const char *str, int affcnt, int (*putfunc) (int));</pre> |
| PARAMETERS | <p><i>str</i> Is a pointer to a terminfo variable or return value from tgetstr(3XCURSES), tgoto(3XCURSES), tigetstr(3XCURSES), or tparm(3XCURSES).</p> <p><i>affcnt</i> Is the number of lines affected, or 1 if not relevant.</p> <p><i>putfunc</i> Is the output function.</p> |
| DESCRIPTION | <p>The putp() and tputs() functions are low-level functions used to deal directly with the terminfo database. The use of appropriate X/Open Curses functions is recommended for most situations.</p> <p>The tputs() function adds padding information and then outputs <i>str</i>. <i>str</i> must be a terminfo string variable or the result value from tgetstr(), tgoto(), tigetstr(), or tparm(). The tputs() function replaces the padding specification (if one exists) with enough characters to produce the specified delay. Characters are output one at a time to <i>putfunc</i>, a user-specified function similar to putchar(3C).</p> <p>The putp() function calls tputs() as follows:</p> <pre>tputs(str, 1, putchar)</pre> |
| RETURN VALUES | On success, these functions return OK. |
| ERRORS | None. |
| USAGE | The output of putp() goes to stdout, not to the file descriptor, <i>fildev</i> , specified in setupterm(3XCURSES). |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: |

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

tputs(3XCURSES)

SEE ALSO libcurses(3XCURSES), putchar(3C), setupterm(3XCURSES),
tgetent(3XCURSES), tigetflag(3XCURSES), terminfo(4), attributes(5),
standards(5)

| | |
|--------------------|---|
| NAME | curs_inopts, cbreak, nocbreak, echo, noecho, halfdelay, intrflush, keypad, meta, nodelay, notimeout, raw, noraw, noqiflush, qiflush, timeout, wtimeout, typeahead – curses terminal input option control routines |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int cbreak(void); int nocbreak(void); int echo(void); int noecho(void); int halfdelay(int tenths); int intrflush(WINDOW *win, bool bf); int keypad(WINDOW *win, bool bf); int meta(WINDOW *win, bool bf); int nodelay(WINDOW *win, bool bf); int notimeout(WINDOW *win, bool bf); int raw(void); int noraw(void); void noqiflush(void); void qiflush(void); void timeout(int delay); void wtimeout(WINDOW *win, int delay); int typeahead(int fildes);</pre> |
| DESCRIPTION | <p>The <code>cbreak()</code> and <code>nocbreak()</code> routines put the terminal into and out of <code>cbreak()</code> mode, respectively. In this mode, characters typed by the user are immediately available to the program, and erase/kill character-processing is not performed. When out of this mode, the tty driver buffers the typed characters until a newline or carriage return is typed. Interrupt and flow control characters are unaffected by this mode. Initially the terminal may or may not be in <code>cbreak()</code> mode, as the mode is inherited; therefore, a program should call <code>cbreak()</code> or <code>nocbreak()</code> explicitly. Most interactive programs using <code>curses</code> set the <code>cbreak()</code> mode.</p> <p>Note that <code>cbreak()</code> overrides <code>raw()</code>. (See <code>curs_getch(3CURSES)</code> for a discussion of how these routines interact with <code>echo()</code> and <code>noecho()</code>.)</p> <p>The <code>echo()</code> and <code>noecho()</code> routines control whether characters typed by the user are echoed by <code>getch()</code> as they are typed. Echoing by the tty driver is always disabled, but initially <code>getch()</code> is in <code>echo</code> mode, so characters typed are echoed. Authors of</p> |

typeahead(3CURSES)

most interactive programs prefer to do their own echoing in a controlled area of the screen, or not to echo at all, so they disable echoing by calling `noecho()`. (See `curs_getch(3CURSES)` for a discussion of how these routines interact with `cbreak()` and `nocbreak()`.)

The `halfdelay()` routine is used for half-delay mode, which is similar to `cbreak()` mode in that characters typed by the user are immediately available to the program. However, after blocking for *tenths* tenths of seconds, `ERR` is returned if nothing has been typed. The value of *tenths* must be a number between 1 and 255. Use `nocbreak()` to leave half-delay mode.

If the `intrflush()` option is enabled, (*bf* is `TRUE`), when an interrupt key is pressed on the keyboard (interrupt, break, quit) all output in the tty driver queue will be flushed, giving the effect of faster response to the interrupt, but causing `curses` to have the wrong idea of what is on the screen. Disabling (*bf* is `FALSE`), the option prevents the flush. The default for the option is inherited from the tty driver settings. The window argument is ignored.

The `keypad()` option enables the keypad of the user's terminal. If enabled (*bf* is `TRUE`), the user can press a function key (such as an arrow key) and `wgetch()` returns a single value representing the function key, as in `KEY_LEFT`. If disabled (*bf* is `FALSE`), `curses` does not treat function keys specially and the program has to interpret the escape sequences itself. If the keypad in the terminal can be turned on (made to transmit) and off (made to work locally), turning on this option causes the terminal keypad to be turned on when `wgetch()` is called. The default value for keypad is false.

Initially, whether the terminal returns 7 or 8 significant bits on input depends on the control mode of the tty driver (see `termio(7I)`). To force 8 bits to be returned, invoke `meta(win, TRUE)`. To force 7 bits to be returned, invoke `meta(win, FALSE)`. The window argument, *win*, is always ignored. If the terminfo capabilities `smm` (`meta_on`) and `rmm` (`meta_off`) are defined for the terminal, `smm` is sent to the terminal when `meta(win, TRUE)` is called and `rmm` is sent when `meta(win, FALSE)` is called.

The `nodelay()` option causes `getch()` to be a non-blocking call. If no input is ready, `getch()` returns `ERR`. If disabled (*bf* is `FALSE`), `getch()` waits until a key is pressed.

While interpreting an input escape sequence, `wgetch()` sets a timer while waiting for the next character. If `notimeout(win, TRUE)` is called, then `wgetch()` does not set a timer. The purpose of the timeout is to differentiate between sequences received from a function key and those typed by a user.

With the `raw()` and `noraw()` routines, the terminal is placed into or out of raw mode. Raw mode is similar to `cbreak()` mode, in that characters typed are immediately passed through to the user program. The differences are that in raw mode, the interrupt, quit, suspend, and flow control characters are all passed through uninterpreted, instead of generating a signal. The behavior of the `BREAK` key depends on other bits in the tty driver that are not set by `curses`.

typeahead(3CURSES)

When the `noqiflush()` routine is used, normal flush of input and output queues associated with the `INTR`, `QUIT` and `SUSP` characters will not be done (see `termio(7I)`). When `qiflush()` is called, the queues will be flushed when these control characters are read.

The `timeout()` and `wtimeout()` routines set blocking or non-blocking read for a given window. If *delay* is negative, blocking read is used (that is, waits indefinitely for input). If *delay* is zero, then non-blocking read is used (that is, read returns `ERR` if no input is waiting). If *delay* is positive, then read blocks for *delay* milliseconds, and returns `ERR` if there is still no input. Hence, these routines provide the same functionality as `nodelay()`, plus the additional capability of being able to block for only *delay* milliseconds (where *delay* is positive).

`curses` does "line-breakout optimization" by looking for `typeahead` periodically while updating the screen. If input is found, and it is coming from a `tty`, the current update is postponed until `refresh()` or `doupdate()` is called again. This allows faster response to commands typed in advance. Normally, the input `FILE` pointer passed to `newterm()`, or `stdin` in the case that `initscr()` was used, will be used to do this `typeahead` checking. The `typeahead()` routine specifies that the file descriptor *fildes* is to be used to check for `typeahead` instead. If *fildes* is `-1`, then no `typeahead` checking is done.

RETURN VALUES

All routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_getch(3CURSES)`, `curs_initscr(3CURSES)`, `curses(3CURSES)`, `attributes(5)`, `termio(7I)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `echo()`, `noecho()`, `halfdelay()`, `intrflush()`, `meta()`, `nodelay()`, `notimeout()`, `noqiflush()`, `qiflush()`, `timeout()`, and `wtimeout()` may be macros.

typeahead(3XCURSES)

| NAME | typeahead – check for type-ahead characters | | | | | | |
|----------------------|--|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int typeahead(int fd);</pre> | | | | | | |
| PARAMETERS | <i>fd</i> Is the file descriptor that is used to check for type-ahead characters. | | | | | | |
| DESCRIPTION | <p>The <code>typeahead()</code> function specifies the file descriptor (<i>fd</i>) to use to check for type-ahead characters (characters typed by the user but not yet processed by X/Open Curses).</p> <p>X/Open Curses checks for type-ahead characters periodically while updating the screen. If characters are found, the current update is postponed until the next <code>refresh(3XCURSES)</code> or <code>doupdate(3XCURSES)</code>. This speeds up response to commands that have been typed ahead. Normally, the input file pointer passed to <code>newterm(3XCURSES)</code>, or <code>stdin</code> in the case of <code>initscr(3XCURSES)</code>, is used for type-ahead checking.</p> <p>If <i>fd</i> is -1, no type-ahead checking is done.</p> | | | | | | |
| RETURN VALUES | On success, the <code>typeahead()</code> function returns OK. Otherwise, it returns ERR. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Standard</td></tr><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>doupdate(3XCURSES)</code> , <code>getch(3XCURSES)</code> , <code>initscr(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

| | |
|--------------------|--|
| NAME | curs_util, unctrl, keyname, filter, use_env, putwin, getwin, delay_output, flushinp – curses miscellaneous utility routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> char *unctrl (chtype <i>c</i>); char *keyname (int <i>c</i>); int filter (void); void use_env (char <i>bool</i>); int putwin (WINDOW *<i>win</i>, FILE *<i>filep</i>); WINDOW *getwin (FILE *<i>filep</i>); int delay_output (int <i>ms</i>); int flushinp (void);</pre> |
| DESCRIPTION | <p>The unctrl () macro expands to a character string which is a printable representation of the character <i>c</i>. Control characters are displayed in the ^X notation. Printing characters are displayed as is.</p> <p>With the keyname () routine, a character string corresponding to the key <i>c</i> is returned.</p> <p>The filter () routine, if used, is called before initscr () or newterm () are called. It makes curses think that there is a one-line screen. curses does not use any terminal capabilities that assume that they know on what line of the screen the cursor is positioned.</p> <p>The use_env () routine, if used, is called before initscr () or newterm () are called. When called with FALSE as an argument, the values of lines and columns specified in the terminfo database will be used, even if environment variables LINES and COLUMNS (used by default) are set, or if curses is running in a window (in which case default behavior would be to use the window size if LINES and COLUMNS are not set).</p> <p>With the putwin () routine, all data associated with window <i>win</i> is written into the file to which <i>filep</i> points. This information can be later retrieved using the getwin () function.</p> <p>The getwin () routine reads window related data stored in the file by putwin (). The routine then creates and initializes a new window using that data. It returns a pointer to the new window.</p> <p>The delay_output () routine inserts an <i>ms</i> millisecond pause in output. This routine should not be used extensively because padding characters are used rather than a CPU pause.</p> <p>The flushinp () routine throws away any typeahead that has been typed by the user and has not yet been read by the program.</p> |

unctrl(3CURSES)

RETURN VALUES Except for `flushinp()`, routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`flushinp()` always returns `OK`.

Routines that return pointers return `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_initscr(3CURSES)`, `curs_scr_dump(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `unctrl()` is a macro, which is defined in `<unctrl.h>`.

NAME unctrl – generate printable representation of a character

SYNOPSIS `cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \`
`-R /usr/xpg4/lib -lcurses [library...]`

`c89 [flag...] file... -lcurses [library...]`

`#include <unctrl.h>`

`char *unctrl (chtype c);`

PARAMETERS `c` Is a character.

DESCRIPTION The unctrl () function generates a character string that is a printable representation of `c`. If `c` is a control character, it is converted to the ^X notation. If `c` contains rendition information, the effect is undefined.

RETURN VALUES Upon successful completion, the unctrl () function returns the generated string. Otherwise, it returns a null pointer.

ERRORS No errors are defined.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO addch(3XCURSES), addstr(3XCURSES), libcurses(3XCURSES),
wunctrl(3XCURSES), attributes(5), standards(5)

ungetch(3CURSES)

| NAME | <code> curs_getch</code> , <code> getch</code> , <code> wgetch</code> , <code> mvgetch</code> , <code> mvwgetch</code> , <code> ungetch</code> – get (or push back) characters from curses terminal keyboard | | | | |
|-------------------------|--|-------------|-----------------|-------------------------|-----------|
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int getch(void); int wgetch(WINDOW *<i>win</i>); int mvgetch(int <i>y</i>, int <i>x</i>); int mvwgetch(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>); int ungetch(int <i>ch</i>);</pre> | | | | |
| DESCRIPTION | <p>With the <code> getch()</code>, <code> wgetch()</code>, <code> mvgetch()</code>, and <code> mvwgetch()</code> routines a character is read from the terminal associated with the window. In no-delay mode, if no input is waiting, the value <code> ERR</code> is returned. In delay mode, the program waits until the system passes text through to the program. Depending on the setting of <code> cbreak()</code>, this is after one character (<code> cbreak</code> mode), or after the first newline (<code> nocbreak</code> mode). In half-delay mode, the program waits until a character is typed or the specified timeout has been reached. Unless <code> noecho()</code> has been set, the character will also be echoed into the designated window.</p> <p>If the window is not a pad, and it has been moved or modified since the last call to <code> wrefresh()</code>, <code> wrefresh()</code> will be called before another character is read.</p> <p>If <code> keypad()</code> is <code> TRUE</code>, and a function key is pressed, the token for that function key is returned instead of the raw characters. Possible function keys are defined in <code> <curses.h></code> with integers beginning with <code> 0401</code>, whose names begin with <code> KEY_</code>. If a character that could be the beginning of a function key (such as escape) is received, <code> curses</code> sets a timer. If the remainder of the sequence does not come in within the designated time, the character is passed through; otherwise, the function key value is returned. For this reason, many terminals experience a delay between the time a user presses the escape key and the escape is returned to the program. Since tokens returned by these routines are outside the ASCII range, they are not printable.</p> <p>The <code> ungetch()</code> routine places <code> ch</code> back onto the input queue to be returned by the next call to <code> wgetch()</code>.</p> | | | | |
| Function Keys | <p>The following function keys, defined in <code> <curses.h></code>, might be returned by <code> getch()</code> if <code> keypad()</code> has been enabled. Note that not all of these may be supported on a particular terminal if the terminal does not transmit a unique code when the key is pressed or if the definition for the key is not present in the <code> terminfo</code> database.</p> <table><thead><tr><th><i>Name</i></th><th><i>Key name</i></th></tr></thead><tbody><tr><td><code> KEY_BREAK</code></td><td>Break key</td></tr></tbody></table> | <i>Name</i> | <i>Key name</i> | <code> KEY_BREAK</code> | Break key |
| <i>Name</i> | <i>Key name</i> | | | | |
| <code> KEY_BREAK</code> | Break key | | | | |

| <i>Name</i> | <i>Key name</i> |
|-------------------|---|
| KEY_DOWN | The four arrow keys . . . |
| KEY_UP | |
| KEY_LEFT | |
| KEY_RIGHT | |
| KEY_HOME | Home key (upward+left arrow) |
| KEY_BACKSPACE | Backspace |
| KEY_F0 | Function keys; space for 64 keys is reserved. |
| KEY_F(<i>n</i>) | For $0 \leq n \leq 63$ |
| KEY_DL | Delete line |
| KEY_IL | Insert line |
| KEY_DC | Delete character |
| KEY_IC | Insert char or enter insert mode |
| KEY_EIC | Exit insert char mode |
| KEY_CLEAR | Clear screen |
| KEY_EOS | Clear to end of screen |
| KEY_EOL | Clear to end of line |
| KEY_SF | Scroll 1 line forward |
| KEY_SR | Scroll 1 line backward (reverse) |
| KEY_NPAGE | Next page |
| KEY_PPAGE | Previous page |
| KEY_STAB | Set tab |
| KEY_CTAB | Clear tab |
| KEY_CATAB | Clear all tabs |
| KEY_ENTER | Enter or send |
| KEY_SRESET | Soft (partial) reset |
| KEY_RESET | Reset or hard reset |
| KEY_PRINT | Print or copy |
| KEY_LL | Home down or bottom (lower left). Keypad is arranged like this: (Row 1) A1 up A3 (Row 2) left B2 right (Row 3) C1 down C3 |

ungetch(3CURSES)

| <i>Name</i> | <i>Key name</i> |
|---------------|-----------------------|
| KEY_A1 | Upper left of keypad |
| KEY_A3 | Upper right of keypad |
| KEY_B2 | Center of keypad |
| KEY_C1 | Lower left of keypad |
| KEY_C3 | Lower right of keypad |
| KEY_BTAB | Back tab key |
| KEY_BEG | Beg(inning) key |
| KEY_CANCEL | Cancel key |
| KEY_CLOSE | Close key |
| KEY_COMMAND | Cmd (command) key |
| KEY_COPY | Copy key |
| KEY_CREATE | Create key |
| KEY_END | End key |
| KEY_EXIT | Exit key |
| KEY_FIND | Find key |
| KEY_HELP | Help key |
| KEY_MARK | Mark key |
| KEY_MESSAGE | Message key |
| KEY_MOVE | Move key |
| KEY_NEXT | Next object key |
| KEY_OPEN | Open key |
| KEY_OPTIONS | Options key |
| KEY_PREVIOUS | Previous object key |
| KEY_REDO | Redo key |
| KEY_REFERENCE | Reference key |
| KEY_REFRESH | Refresh key |
| KEY_REPLACE | Replace key |
| KEY_RESTART | Restart key |
| KEY_RESUME | Resume key |

| <i>Name</i> | <i>Key name</i> |
|---------------|-------------------------|
| KEY_SAVE | Save key |
| KEY_SBEG | Shifted beginning key |
| KEY_SCANCEL | Shifted cancel key |
| KEY_SCOMMAND | Shifted command key |
| KEY_SCOPY | Shifted copy key |
| KEY_SCREATE | Shifted create key |
| KEY_SDC | Shifted delete char key |
| KEY_SDL | Shifted delete line key |
| KEY_SELECT | Select key |
| KEY_SEND | Shifted end key |
| KEY_SEOL | Shifted clear line key |
| KEY_SEXIT | Shifted exit key |
| KEY_SFIND | Shifted find key |
| KEY_SHELP | Shifted help key |
| KEY_SHOME | Shifted home key |
| KEY_SIC | Shifted input key |
| KEY_SLEFT | Shifted left arrow key |
| KEY_SMESSAGE | Shifted message key |
| KEY_SMOVE | Shifted move key |
| KEY_SNEXT | Shifted next key |
| KEY_SOPTIONS | Shifted options key |
| KEY_SPREVIOUS | Shifted prev key |
| KEY_SPRINT | Shifted print key |
| KEY_SREDO | Shifted redo key |
| KEY_SREPLACE | Shifted replace key |
| KEY_SRIGHT | Shifted right arrow |
| KEY_SRSUME | Shifted resume key |
| KEY_SSAVE | Shifted save key |
| KEY_SSUSPEND | Shifted suspend key |

ungetch(3CURSES)

| <i>Name</i> | <i>Key name</i> |
|-------------|------------------|
| KEY_SUNDO | Shifted undo key |
| KEY_SUSPEND | Suspend key |
| KEY_UNDO | Undo key |

RETURN VALUES All routines return the integer `ERR` upon failure. The `ungetch()` routine returns an integer value other than `ERR` upon successful completion. The other routines return the next input character or function key code upon successful completion.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|-----------------------|------------------------|
| MT-Level | Unsafe |

SEE ALSO `curs_inopts(3CURSES)`, `curs_move(3CURSES)`, `curs_refresh(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Use of the escape key for a single character function is discouraged.

When using `getch()`, `wgetch()`, `mvgetch()`, or `mvwgetch()`, `nocbreak` mode (`nocbreak()`) and `echo` mode (`echo()`) should not be used at the same time. Depending on the state of the tty driver when each character is typed, the program may produce undesirable results.

Note that `getch()`, `mvgetch()`, and `mvwgetch()` may be macros.

| NAME | ungetch, unget_wch – push character back onto the input queue | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int ungetch(int ch); int unget_wch(const wchar_t wch);</pre> | | | | | | |
| PARAMETERS | <p><i>ch</i> Is the single byte character to be put back in the input queue for the next call to <code>getch(3XCURSES)</code>.</p> <p><i>wch</i> Is the wide character to be put back in the input queue for the next call to <code>get_wch(3XCURSES)</code>.</p> | | | | | | |
| DESCRIPTION | <p>The <code>ungetch()</code> function pushes <i>ch</i> back onto the input queue until the next call to <code>getch()</code>.</p> <p>The <code>unget_wch()</code> function is similar to <code>ungetch()</code> except that <i>ch</i> can be of type <code>wchar_t</code>.</p> | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>get_wch(3XCURSES)</code> , <code>getch(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

ungetwch(3CURSES)

| NAME | <code>curs_getwch</code> , <code>getwch</code> , <code>wgetwch</code> , <code>mvgetwch</code> , <code>mvwgetwch</code> , <code>ungetwch</code> – get (or push back) <code>wchar_t</code> characters from curses terminal keyboard | | | | |
|------------------------|--|------|----------|------------------------|-----------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int getwch(void); int wgetwch(WINDOW *win); int mvgetwch(int y, int x); int mvwgetwch(WINDOW *win, int y, int x); int ungetwch(int wch);</pre> | | | | |
| DESCRIPTION | <p>The <code>getwch()</code>, <code>wgetwch()</code>, <code>mvgetwch()</code>, and <code>mvwgetwch()</code> routines read an EUC character from the terminal associated with the window, transform it into a <code>wchar_t</code> character, and return a <code>wchar_t</code> character. In no-delay mode, if no input is waiting, the value <code>ERR</code> is returned. In delay mode, the program waits until the system passes text through to the program. Depending on the setting of <code>cbreak</code>, this is after one character (<code>cbreak</code> mode), or after the first newline (<code>nocbreak</code> mode). In half-delay mode, the program waits until a character is typed or the specified timeout has been reached. Unless <code>noecho</code> has been set, the character will also be echoed into the designated window.</p> <p>If the window is not a pad, and it has been moved or modified since the last call to <code>wrefresh(3CURSES)</code>, <code>wrefresh</code> will be called before another character is read.</p> <p>If <code>keypad</code> is <code>TRUE</code>, and a function key is pressed, the token for that function key is returned instead of the raw characters. Possible function keys are defined in <code><curses.h></code> with integers beginning with <code>0401</code>, whose names begin with <code>KEY_</code>. If a character that could be the beginning of a function key (such as escape) is received, <code>curses(3CURSES)</code> sets a timer. If the remainder of the sequence does not come in within the designated time, the character is passed through; otherwise, the function key value is returned. For this reason, many terminals experience a delay between the time a user presses the escape key and the escape is returned to the program.</p> <p>The <code>ungetwch()</code> routine places <code>wch</code> back onto the input queue to be returned by the next call to <code>wgetwch()</code>.</p> | | | | |
| Function Keys | <p>The following function keys, defined in <code><curses.h></code>, might be returned by <code>getwch()</code> if <code>keypad</code> has been enabled. Note that not all of these may be supported on a particular terminal if the terminal does not transmit a unique code when the key is pressed or if the definition for the key is not present in the <code>terminfo(4)</code> database.</p> <table border="1"><thead><tr><th>Name</th><th>Key name</th></tr></thead><tbody><tr><td><code>KEY_BREAK</code></td><td>Break key</td></tr></tbody></table> | Name | Key name | <code>KEY_BREAK</code> | Break key |
| Name | Key name | | | | |
| <code>KEY_BREAK</code> | Break key | | | | |

| <i>Name</i> | <i>Key name</i> |
|-------------------|---|
| KEY_DOWN | The four arrow keys . . . |
| KEY_UP | |
| KEY_LEFT | |
| KEY_RIGHT | |
| KEY_HOME | Home key (upward+left arrow) |
| KEY_BACKSPACE | Backspace |
| KEY_F0 | Function keys; space for 64 keys is reserved. |
| KEY_F(<i>n</i>) | For $0 \leq n \leq 63$ |
| KEY_DL | Delete line |
| KEY_IL | Insert line |
| KEY_DC | Delete character |
| KEY_IC | Insert char or enter insert mode |
| KEY_EIC | Exit insert char mode |
| KEY_CLEAR | Clear screen |
| KEY_EOS | Clear to end of screen |
| KEY_EOL | Clear to end of line |
| KEY_SF | Scroll 1 line forward |
| KEY_SR | Scroll 1 line backward (reverse) |
| KEY_NPAGE | Next page |
| KEY_PPAGE | Previous page |
| KEY_STAB | Set tab |
| KEY_CTAB | Clear tab |
| KEY_CATAB | Clear all tabs |
| KEY_ENTER | Enter or send |
| KEY_SRESET | Soft (partial) reset |
| KEY_RESET | Reset or hard reset |
| KEY_PRINT | Print or copy |
| KEY_LL | Home down or bottom (lower left). Keypad is arranged like this: A1 up A3 left B2 right C1 down C3 |

ungetwch(3CURSES)

| <i>Name</i> | <i>Key name</i> |
|---------------|-----------------------|
| KEY_A1 | Upper left of keypad |
| KEY_A3 | Upper right of keypad |
| KEY_B2 | Center of keypad |
| KEY_C1 | Lower left of keypad |
| KEY_C3 | Lower right of keypad |
| KEY_BTAB | Back tab key |
| KEY_BEG | Beg(inning) key |
| KEY_CANCEL | Cancel key |
| KEY_CLOSE | Close key |
| KEY_COMMAND | Cmd (command) key |
| KEY_COPY | Copy key |
| KEY_CREATE | Create key |
| KEY_END | End key |
| KEY_EXIT | Exit key |
| KEY_FIND | Find key |
| KEY_HELP | Help key |
| KEY_MARK | Mark key |
| KEY_MESSAGE | Message key |
| KEY_MOVE | Move key |
| KEY_NEXT | Next object key |
| KEY_OPEN | Open key |
| KEY_OPTIONS | Options key |
| KEY_PREVIOUS | Previous object key |
| KEY_REDO | Redo key |
| KEY_REFERENCE | Reference key |
| KEY_REFRESH | Refresh key |
| KEY_REPLACE | Replace key |
| KEY_RESTART | Restart key |
| KEY_RESUME | Resume key |

| <i>Name</i> | <i>Key name</i> |
|---------------|-------------------------|
| KEY_SAVE | Save key |
| KEY_SBEG | Shifted beginning key |
| KEY_SCANCEL | Shifted cancel key |
| KEY_SCOMMAND | Shifted command key |
| KEY_SCOPY | Shifted copy key |
| KEY_SCREATE | Shifted create key |
| KEY_SDC | Shifted delete char key |
| KEY_SDL | Shifted delete line key |
| KEY_SELECT | Select key |
| KEY_SEND | Shifted end key |
| KEY_SEOL | Shifted clear line key |
| KEY_SEXIT | Shifted exit key |
| KEY_SFIND | Shifted find key |
| KEY_SHELP | Shifted help key |
| KEY_SHOME | Shifted home key |
| KEY_SIC | Shifted input key |
| KEY_SLEFT | Shifted left arrow key |
| KEY_SMESSAGE | Shifted message key |
| KEY_SMOVE | Shifted move key |
| KEY_SNEXT | Shifted next key |
| KEY_SOPTIONS | Shifted options key |
| KEY_SPREVIOUS | Shifted prev key |
| KEY_SPRINT | Shifted print key |
| KEY_SREDO | Shifted redo key |
| KEY_SREPLACE | Shifted replace key |
| KEY_SRIGHT | Shifted right arrow |
| KEY_SRSUME | Shifted resume key |
| KEY_SSAVE | Shifted save key |
| KEY_SSUSPEND | Shifted suspend key |

ungetwch(3CURSES)

| <i>Name</i> | <i>Key name</i> |
|-------------|------------------|
| KEY_SUNDO | Shifted undo key |
| KEY_SUSPEND | Suspend key |
| KEY_UNDO | Undo key |

RETURN VALUE All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3CURSES)`, `curl_inopts(3CURSES)`, `curl_move(3CURSES)`, `wrefresh(3CURSES)`, `terminfo(4)`, `attributes(5)`

NOTES The header file `<curses.h>` automatically includes the header files `<stdio.h>`, `<unctrl.h>` and `<widec.h>`.

Use of the escape key by a programmer for a single character function is discouraged.

When using `getwch()`, `wgetwch()`, `mvgetwch()`, or `mvwgetwch()`, `nocbreak` mode and `echo` mode should not be used at the same time. Depending on the state of the tty driver when each character is typed, the program may produce undesirable results.

Note that `getwch()`, `mvgetwch()`, and `mvwgetwch()` may be macros.

NAME ungetch, unget_wch – push character back onto the input queue

SYNOPSIS `cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \`
`-R /usr/xpg4/lib -lcurses [library...]`

`c89 [flag...] file... -lcurses [library...]`

```
#include <curses.h>
int ungetch(int ch);
int unget_wch(const wchar_t wch);
```

PARAMETERS *ch* Is the single byte character to be put back in the input queue for the next call to `getch(3XCURSES)`.

wch Is the wide character to be put back in the input queue for the next call to `get_wch(3XCURSES)`.

DESCRIPTION The `ungetch()` function pushes *ch* back onto the input queue until the next call to `getch()`.

The `unget_wch()` function is similar to `ungetch()` except that *ch* can be of type `wchar_t`.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `get_wch(3XCURSES)`, `getch(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

unpost_form(3CURSES)

| NAME | form_post, post_form, unpost_form – write or erase forms from associated subwindows | | | | | | | | | | | | | | | | |
|----------------------|--|----------------|-------------------------------------|----------------|---------------|----------------|---------------------------|----------|---------------------|--------------|-------------------------|-----------|---|-------------|--|-----------------|---------------------------------------|
| SYNOPSIS | <pre>cc [flag...] file... -lform -lcurses [library...] #include <form.h> int post_form(FORM *form) ; int unpost_form(FORM *form) ;</pre> | | | | | | | | | | | | | | | | |
| DESCRIPTION | <p>post_form() writes <i>form</i> into its associated subwindow. The application programmer must use curses library routines to display the form on the physical screen or call update_panels() if the panels library is being used.</p> <p>unpost_form() erases <i>form</i> from its associated subwindow.</p> | | | | | | | | | | | | | | | | |
| RETURN VALUES | <p>These routines return one of the following:</p> <table><tr><td>E_OK</td><td>The function returned successfully.</td></tr><tr><td>E_SYSTEM_ERROR</td><td>System error.</td></tr><tr><td>E_BAD_ARGUMENT</td><td>An argument is incorrect.</td></tr><tr><td>E_POSTED</td><td>The form is posted.</td></tr><tr><td>E_NOT_POSTED</td><td>The form is not posted.</td></tr><tr><td>E_NO_ROOM</td><td>The form does not fit in the subwindow.</td></tr><tr><td>E_BAD_STATE</td><td>The routine was called from an initialization or termination function.</td></tr><tr><td>E_NOT_CONNECTED</td><td>The field is not connected to a form.</td></tr></table> | E_OK | The function returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An argument is incorrect. | E_POSTED | The form is posted. | E_NOT_POSTED | The form is not posted. | E_NO_ROOM | The form does not fit in the subwindow. | E_BAD_STATE | The routine was called from an initialization or termination function. | E_NOT_CONNECTED | The field is not connected to a form. |
| E_OK | The function returned successfully. | | | | | | | | | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | | | | | | | | | |
| E_BAD_ARGUMENT | An argument is incorrect. | | | | | | | | | | | | | | | | |
| E_POSTED | The form is posted. | | | | | | | | | | | | | | | | |
| E_NOT_POSTED | The form is not posted. | | | | | | | | | | | | | | | | |
| E_NO_ROOM | The form does not fit in the subwindow. | | | | | | | | | | | | | | | | |
| E_BAD_STATE | The routine was called from an initialization or termination function. | | | | | | | | | | | | | | | | |
| E_NOT_CONNECTED | The field is not connected to a form. | | | | | | | | | | | | | | | | |
| ATTRIBUTES | <p>See attributes(5) for descriptions of the following attributes:</p> <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | | | | | | | | | | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | | | | | | | | | | | |
| MT-Level | Unsafe | | | | | | | | | | | | | | | | |
| SEE ALSO | curses(3CURSES), forms(3CURSES), panel_update(3CURSES), panels(3CURSES), attributes(5) | | | | | | | | | | | | | | | | |
| NOTES | The header <form.h> automatically includes the headers <eti.h> and <curses.h>. | | | | | | | | | | | | | | | | |

| NAME | menu_post, post_menu, unpost_menu – write or erase menus from associated subwindows | | | | | | | | | | | | | | | | |
|----------------------|--|----------------|------------------------------------|----------------|---------------|----------------|--|----------|-----------------------------|-------------|--|-----------|---|--------------|-------------------------------|-----------------|-------------------------------------|
| SYNOPSIS | <pre>cc [flag...] file... -lmenu -lcurses [library...] #include <menu.h> int post_menu(MENU *menu); int unpost_menu(MENU *menu);</pre> | | | | | | | | | | | | | | | | |
| DESCRIPTION | <p>post_menu() writes <i>menu</i> to the subwindow. The application programmer must use curses library routines to display the menu on the physical screen or call update_panels() if the panels library is being used.</p> <p>unpost_menu() erases <i>menu</i> from its associated subwindow.</p> | | | | | | | | | | | | | | | | |
| RETURN VALUES | <p>These routines return one of the following:</p> <table border="0"> <tr> <td>E_OK</td> <td>The routine returned successfully.</td> </tr> <tr> <td>E_SYSTEM_ERROR</td> <td>System error.</td> </tr> <tr> <td>E_BAD_ARGUMENT</td> <td>An incorrect argument was passed to the routine.</td> </tr> <tr> <td>E_POSTED</td> <td>The menu is already posted.</td> </tr> <tr> <td>E_BAD_STATE</td> <td>The routine was called from an initialization or termination function.</td> </tr> <tr> <td>E_NO_ROOM</td> <td>The menu does not fit within its subwindow.</td> </tr> <tr> <td>E_NOT_POSTED</td> <td>The menu has not been posted.</td> </tr> <tr> <td>E_NOT_CONNECTED</td> <td>No items are connected to the menu.</td> </tr> </table> | E_OK | The routine returned successfully. | E_SYSTEM_ERROR | System error. | E_BAD_ARGUMENT | An incorrect argument was passed to the routine. | E_POSTED | The menu is already posted. | E_BAD_STATE | The routine was called from an initialization or termination function. | E_NO_ROOM | The menu does not fit within its subwindow. | E_NOT_POSTED | The menu has not been posted. | E_NOT_CONNECTED | No items are connected to the menu. |
| E_OK | The routine returned successfully. | | | | | | | | | | | | | | | | |
| E_SYSTEM_ERROR | System error. | | | | | | | | | | | | | | | | |
| E_BAD_ARGUMENT | An incorrect argument was passed to the routine. | | | | | | | | | | | | | | | | |
| E_POSTED | The menu is already posted. | | | | | | | | | | | | | | | | |
| E_BAD_STATE | The routine was called from an initialization or termination function. | | | | | | | | | | | | | | | | |
| E_NO_ROOM | The menu does not fit within its subwindow. | | | | | | | | | | | | | | | | |
| E_NOT_POSTED | The menu has not been posted. | | | | | | | | | | | | | | | | |
| E_NOT_CONNECTED | No items are connected to the menu. | | | | | | | | | | | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | | | | | | | | | | | |
| | <table border="1"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | | | | | | | | | | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | | | | | | | | | | | |
| MT-Level | Unsafe | | | | | | | | | | | | | | | | |
| SEE ALSO | curses(3CURSES), menus(3CURSES), panels(3CURSES), attributes(5) | | | | | | | | | | | | | | | | |
| NOTES | The header <menu.h> automatically includes the headers <eti.h> and <curses.h>. | | | | | | | | | | | | | | | | |

untouchwin(3CURSES)

| NAME | <code> curs_touch, touchwin, touchline, untouchwin, wtouchln, is_linetouched, is_wintouched</code> – curses refresh control routines | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int touchwin(WINDOW *win); int touchline(WINDOW *win, int start, int count); int untouchwin(WINDOW *win); int wtouchln(WINDOW *win, int y, int n, int changed); int is_linetouched(WINDOW *win, int line); int is_wintouched(WINDOW *win);</pre> | | | | |
| DESCRIPTION | <p>The <code>touchwin()</code> and <code>touchline()</code> routines throw away all optimization information about which parts of the window have been touched, by pretending that the entire window has been drawn on. This is sometimes necessary when using overlapping windows, since a change to one window affects the other window, but the records of which lines have been changed in the other window do not reflect the change. The routine <code>touchline()</code> only pretends that <code>count</code> lines have been changed, beginning with line <code>start</code>.</p> <p>The <code>untouchwin()</code> routine marks all lines in the window as unchanged since the last call to <code>wrefresh()</code>.</p> <p>The <code>wtouchln()</code> routine makes <code>n</code> lines in the window, starting at line <code>y</code>, look as if they have (<code>changed=1</code>) or have not (<code>changed=0</code>) been changed since the last call to <code>wrefresh()</code>.</p> <p>The <code>is_linetouched()</code> and <code>is_wintouched()</code> routines return TRUE if the specified line/window was modified since the last call to <code>wrefresh()</code>; otherwise they return FALSE. In addition, <code>is_linetouched()</code> returns ERR if line is not valid for the given window.</p> | | | | |
| RETURN VALUES | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion, unless otherwise noted in the preceding routine descriptions. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_refresh(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . | | | | |

`untouchwin(3CURSES)`

Note that all routines except `wtouchln()` may be macros.

untouchwin(3XCURSES)

| | | | | | | | | | | | | | | | |
|--------------------|--|------------|---|-------------|---|--------------|--|--------------|--|----------|---|----------|--|----------------|---|
| NAME | <code>is_linetouched</code> , <code>is_wintouched</code> , <code>touchline</code> , <code>touchwin</code> , <code>untouchwin</code> , <code>wtouchln</code> – control window refresh | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> bool is_linetouched(WINDOW *win, int line); bool is_wintouched(WINDOW *win); int touchline(WINDOW *win, int start, int count); int touchwin(WINDOW *win); int untouchwin(WINDOW *win); int wtouchln(WINDOW *win, int y, int n, int changed);</pre> | | | | | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>win</i></td><td>Is a pointer to the window in which the refresh is to be controlled or monitored.</td></tr><tr><td><i>line</i></td><td>Is the line to be checked for change since refresh.</td></tr><tr><td><i>start</i></td><td>Is the starting line number of the portion of the window to make appear changed.</td></tr><tr><td><i>count</i></td><td>Is the number of lines in the window to mark as changed.</td></tr><tr><td><i>y</i></td><td>Is the starting line number of the portion of the window to make appear changed or not changed.</td></tr><tr><td><i>n</i></td><td>Is the number of lines in the window to mark as changed.</td></tr><tr><td><i>changed</i></td><td>Is a flag indicating whether to make lines look changed (0) or not changed (1).</td></tr></table> | <i>win</i> | Is a pointer to the window in which the refresh is to be controlled or monitored. | <i>line</i> | Is the line to be checked for change since refresh. | <i>start</i> | Is the starting line number of the portion of the window to make appear changed. | <i>count</i> | Is the number of lines in the window to mark as changed. | <i>y</i> | Is the starting line number of the portion of the window to make appear changed or not changed. | <i>n</i> | Is the number of lines in the window to mark as changed. | <i>changed</i> | Is a flag indicating whether to make lines look changed (0) or not changed (1). |
| <i>win</i> | Is a pointer to the window in which the refresh is to be controlled or monitored. | | | | | | | | | | | | | | |
| <i>line</i> | Is the line to be checked for change since refresh. | | | | | | | | | | | | | | |
| <i>start</i> | Is the starting line number of the portion of the window to make appear changed. | | | | | | | | | | | | | | |
| <i>count</i> | Is the number of lines in the window to mark as changed. | | | | | | | | | | | | | | |
| <i>y</i> | Is the starting line number of the portion of the window to make appear changed or not changed. | | | | | | | | | | | | | | |
| <i>n</i> | Is the number of lines in the window to mark as changed. | | | | | | | | | | | | | | |
| <i>changed</i> | Is a flag indicating whether to make lines look changed (0) or not changed (1). | | | | | | | | | | | | | | |
| DESCRIPTION | <p>The <code>touchwin()</code> function marks the entire window as dirty. This makes it appear to X/Open Curses as if the whole window has been changed, thus causing the entire window to be rewritten with the next call to <code>refresh(3XCURSES)</code>. This is sometimes necessary when using overlapping windows; the change to one window will not be reflected in the other and, hence will not be recorded.</p> <p>The <code>touchline()</code> function marks as dirty a portion of the window starting at line <i>start</i> and continuing for <i>count</i> lines instead of the entire window. Consequently, that portion of the window is updated with the next call to <code>refresh()</code>.</p> <p>The <code>untouchwin()</code> function marks all lines in the window as unchanged since the last refresh, ensuring that it is not updated.</p> | | | | | | | | | | | | | | |

untouchwin(3XCURSES)

The `wtouchln()` function marks n lines starting at line y as either changed ($changed=1$) or unchanged ($changed=0$) since the last refresh.

To find out which lines or windows have been changed since the last refresh, use the `is_linetouched()` and `is_wintouched()` functions, respectively. These return `TRUE` if the specified line or window have been changed since the last call to `refresh()` or `FALSE` if no changes have been made.

RETURN VALUES On success, these functions return `OK`. Otherwise, they return `ERR`.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `doupdate(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

update_panels(3CURSES)

| NAME | panel_update, update_panels – panels virtual screen refresh routine | | | | |
|--------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lpanel -lcurses [library ..] #include <panel.h> void update_panels(void);</pre> | | | | |
| DESCRIPTION | update_panels() refreshes the virtual screen to reflect the depth relationships between the panels in the deck. The user must use the curses library call douupdate() (see curs_refresh(3CURSES)) to refresh the physical screen. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curs_refresh(3CURSES), curses(3CURSES), panels(3CURSES), attributes(5) | | | | |
| NOTES | The header <panel.h> automatically includes the header <curses.h>. | | | | |

| | |
|--------------------|--|
| NAME | curs_util, unctrl, keyname, filter, use_env, putwin, getwin, delay_output, flushinp – curses miscellaneous utility routines |
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> char *unctrl (chtype <i>c</i>); char *keyname (int <i>c</i>); int filter (void); void use_env (char <i>bool</i>); int putwin (WINDOW *<i>win</i>, FILE *<i>filep</i>); WINDOW *getwin (FILE *<i>filep</i>); int delay_output (int <i>ms</i>); int flushinp (void);</pre> |
| DESCRIPTION | <p>The <code>unctrl()</code> macro expands to a character string which is a printable representation of the character <code>c</code>. Control characters are displayed in the <code>^X</code> notation. Printing characters are displayed as is.</p> <p>With the <code>keyname()</code> routine, a character string corresponding to the key <code>c</code> is returned.</p> <p>The <code>filter()</code> routine, if used, is called before <code>initscr()</code> or <code>newterm()</code> are called. It makes <code>curses</code> think that there is a one-line screen. <code>curses</code> does not use any terminal capabilities that assume that they know on what line of the screen the cursor is positioned.</p> <p>The <code>use_env()</code> routine, if used, is called before <code>initscr()</code> or <code>newterm()</code> are called. When called with <code>FALSE</code> as an argument, the values of <code>lines</code> and <code>columns</code> specified in the <code>terminfo</code> database will be used, even if environment variables <code>LINES</code> and <code>COLUMNS</code> (used by default) are set, or if <code>curses</code> is running in a window (in which case default behavior would be to use the window size if <code>LINES</code> and <code>COLUMNS</code> are not set).</p> <p>With the <code>putwin()</code> routine, all data associated with window <code>win</code> is written into the file to which <code>filep</code> points. This information can be later retrieved using the <code>getwin()</code> function.</p> <p>The <code>getwin()</code> routine reads window related data stored in the file by <code>putwin()</code>. The routine then creates and initializes a new window using that data. It returns a pointer to the new window.</p> <p>The <code>delay_output()</code> routine inserts an <code>ms</code> millisecond pause in output. This routine should not be used extensively because padding characters are used rather than a CPU pause.</p> <p>The <code>flushinp()</code> routine throws away any typeahead that has been typed by the user and has not yet been read by the program.</p> |

use_env(3CURSES)

RETURN VALUES Except for `flushinp()`, routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`flushinp()` always returns `OK`.

Routines that return pointers return `NULL` on error.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curs_initscr(3CURSES)`, `curs_scr_dump(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `unctrl()` is a macro, which is defined in `<unctrl.h>`.

| NAME | use_env – specify source of screen size information | | | | | | |
|----------------------|--|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> void use_env (bool <i>boolval</i>);</pre> | | | | | | |
| PARAMETERS | <i>boolval</i> Is a Boolean expression. | | | | | | |
| DESCRIPTION | <p>The use_env() function specifies the technique by which the implementation determines the size of the screen. If <i>boolval</i> is FALSE, the implementation uses the values of <i>lines</i> and <i>columns</i> specified in the terminfo database. If <i>boolval</i> is TRUE, the implementation uses the LINES and COLUMNS environmental variables. The initial value is TRUE.</p> <p>Any call to use_env() must precede calls to initscr(3XCURSES), newterm(3XCURSES), or setupterm(3XCURSES).</p> | | | | | | |
| RETURN VALUES | The use_env() function does not return a value. | | | | | | |
| ERRORS | No errors are defined. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | del_curterm(3XCURSES), initscr(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5) | | | | | | |

vidattr(3CURSES)

| | |
|--------------------|---|
| NAME | <code>curs_terminfo</code> , <code>setupterm</code> , <code>setterm</code> , <code>set_curterm</code> , <code>del_curterm</code> , <code>restartterm</code> , <code>tparam</code> , <code>tputs</code> , <code>putp</code> , <code>vidputs</code> , <code>vidattr</code> , <code>mvcur</code> , <code>tigetflag</code> , <code>tigetnum</code> , <code>tigetstr</code> – curses interfaces to terminfo database |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> #include <term.h> int setupterm(char *term, int fildes, int *errret); int setterm(char *term); int set_curterm(TERMINAL *nterm); int del_curterm(TERMINAL *oterm); int restartterm(char *term, int fildes, int *errret); char *tparam(char *str, long int p1, long int p2, long int p3, long int p4, long int p5, long int p6, long int p7, long int p8, long int p9); int tputs(char *str, int affcnt, int (*putc)(char)); int putp(char *str); int vidputs(chtype attrs, int (*putc)(char)); int vidattr(chtype attrs); int mvcur(int oldrow, int oldcol, int newrow, int newcol); int tigetflag(char *capname); int tigetnum(char *capname); char *tigetstr(char *capname);</pre> |
| DESCRIPTION | <p>These low-level routines must be called by programs that have to deal directly with the <i>terminfo</i> database to handle certain terminal capabilities, such as programming function keys. For all other functionality, curses routines are more suitable and their use is recommended.</p> <p>Initially, <code>setupterm()</code> should be called. Note that <code>setupterm()</code> is automatically called by <code>initscr()</code> and <code>newterm()</code>. This defines the set of terminal-dependent variables (listed in <code>terminfo(4)</code>). The <i>terminfo</i> variables <code>lines</code> and <code>columns</code> are initialized by <code>setupterm()</code> as follows: If <code>use_env(FALSE)</code> has been called, values for <code>lines</code> and <code>columns</code> specified in <i>terminfo</i> are used. Otherwise, if the environment variables <code>LINES</code> and <code>COLUMNS</code> exist, their values are used. If these environment variables do not exist and the program is running in a window, the current window size is used. Otherwise, if the environment variables do not exist, the values for <code>lines</code> and <code>columns</code> specified in the <i>terminfo</i> database are used.</p> <p>The headers <code><curses.h></code> and <code><term.h></code> should be included (in this order) to get the definitions for these strings, numbers, and flags. Parameterized strings should be passed through <code>tparam()</code> to instantiate them. All <i>terminfo</i> strings (including the output</p> |

of `tparm()` should be printed with `tputs()` or `putp()`. Call the `reset_shell_mode()` routine to restore the tty modes before exiting (see `curs_kernel(3CURSES)`). Programs which use cursor addressing should output `enter_ca_mode` upon startup and should output `exit_ca_mode` before exiting. Programs desiring shell escapes should call `reset_shell_mode` and output `exit_ca_mode` before the shell is called and should output `enter_ca_mode` and call `reset_prog_mode` after returning from the shell.

The `setupterm()` routine reads in the *terminfo* database, initializing the *terminfo* structures, but does not set up the output virtualization structures used by *curses*. The terminal type is the character string *term*; if *term* is null, the environment variable `TERM` is used. All output is to file descriptor *fildev* which is initialized for output. If *errret* is not null, then `setupterm()` returns `OK` or `ERR` and stores a status value in the integer pointed to by *errret*. A status of 1 in *errret* is normal, 0 means that the terminal could not be found, and -1 means that the *terminfo* database could not be found. If *errret* is null, `setupterm()` prints an error message upon finding an error and exits. Thus, the simplest call is:

```
setupterm((char *)0, 1, (int *)0);
```

which uses all the defaults and sends the output to `stdout`.

The `setterm()` routine is being replaced by `setupterm()`. The call:

```
setupterm(term, 1, (int *)0)
```

provides the same functionality as `setterm(term)`. The `setterm()` routine is included here for compatibility and is supported at Level 2.

The `set_curterm()` routine sets the variable `cur_term` to *nterm*, and makes all of the *terminfo* boolean, numeric, and string variables use the values from *nterm*.

The `del_curterm()` routine frees the space pointed to by *oterm* and makes it available for further use. If *oterm* is the same as `cur_term`, references to any of the *terminfo* boolean, numeric, and string variables thereafter may refer to invalid memory locations until another `setupterm()` has been called.

The `restartterm()` routine is similar to `setupterm()` and `initscr()`, except that it is called after restoring memory to a previous state. It assumes that the windows and the input and output options are the same as when memory was saved, but the terminal type and baud rate may be different.

The `tparm()` routine instantiates the string *str* with parameters *pi*. A pointer is returned to the result of *str* with the parameters applied.

The `tputs()` routine applies padding information to the string *str* and outputs it. The *str* must be a *terminfo* string variable or the return value from `tparm()`, `tgetstr()`, or `tgoto()`. *affcnt* is the number of lines affected, or 1 if not applicable. *putc* is a `putchar()`-like routine to which the characters are passed, one at a time.

vidattr(3CURSES)

The `putp()` routine calls `tputs(str, 1, putchar)`. Note that the output of `putpA()` always goes to `stdout`, not to the *files* specified in `setupterm()`.

The `vidputs()` routine displays the string on the terminal in the video attribute mode *attrs*, which is any combination of the attributes listed in `curses(3CURSES)`. The characters are passed to the `putchar()`-like routine `putc()`.

The `vidattr()` routine is like the `vidputs()` routine, except that it outputs through `putchar()`.

The `mvcur()` routine provides low-level cursor motion.

The `tigetflag()`, `tigetnum()` and `tigetstr()` routines return the value of the capability corresponding to the *terminfo capname* passed to them, such as `xenl`.

With the `tigetflag()` routine, the value `-1` is returned if *capname* is not a boolean capability.

With the `tigetnum()` routine, the value `-2` is returned if *capname* is not a numeric capability.

With the `tigetstr()` routine, the value `(char *)-1` is returned if *capname* is not a string capability.

The *capname* for each capability is given in the table column entitled *capname code* in the capabilities section of `terminfo(4)`.

```
char *boolnames, *boolcodes, *boolfnames
char *numnames, *numcodes, *numfnames
char *strnames, *strcodes, *strfnames
```

These null-terminated arrays contain the *capnames*, the *termcap* codes, and the full C names, for each of the *terminfo* variables.

RETURN VALUES

All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

Routines that return pointers always return `NULL` on error.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_initscr(3CURSES)`, `curs_kernel(3CURSES)`, `curs_termcap(3CURSES)`, `curses(3CURSES)`, `putc(3C)`, `terminfo(4)`, `attributes(5)`

NOTES | The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

The `setupterm()` routine should be used in place of `setterm()`.

Note that `vidattr()` and `vidputs()` may be macros.

vid_attr(3XCURSES)

| | | | | | | | | | |
|--------------------------|--|-------------|--|--------------------------|------------------|------------|---|----------------|-------------------------------------|
| NAME | vidattr, vid_attr, vidputs, vid_puts – output attributes to the terminal | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int vidattr(chtype attr); int vid_attr(attr_t attr, short color_pair_number, void *opt); int vidputs(chtype attr, int (*putfunc) (int)); int vid_puts(attr_t attr, short color_pair_number, void *opt, int (*putfunc) (int));</pre> | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>attr</i></td> <td>Is the rendition of the foreground window.</td> </tr> <tr> <td><i>color_pair_number</i></td> <td>Is a color pair.</td> </tr> <tr> <td><i>opt</i></td> <td>Is reserved for future use. Currently, this must be a null pointer.</td> </tr> <tr> <td><i>putfunc</i></td> <td>Is a user-supplied output function.</td> </tr> </table> | <i>attr</i> | Is the rendition of the foreground window. | <i>color_pair_number</i> | Is a color pair. | <i>opt</i> | Is reserved for future use. Currently, this must be a null pointer. | <i>putfunc</i> | Is a user-supplied output function. |
| <i>attr</i> | Is the rendition of the foreground window. | | | | | | | | |
| <i>color_pair_number</i> | Is a color pair. | | | | | | | | |
| <i>opt</i> | Is reserved for future use. Currently, this must be a null pointer. | | | | | | | | |
| <i>putfunc</i> | Is a user-supplied output function. | | | | | | | | |
| DESCRIPTION | <p>These functions output commands to the terminal that change the terminal's attributes.</p> <p>If the terminfo database indicates that the terminal in use can display characters in the rendition specified by <i>attr</i>, then <code>vidattr()</code> outputs one or more commands to request that the terminal display subsequent characters in that rendition. The function outputs by calling <code>putchar(3C)</code>. The <code>vidattr()</code> function neither relies on your updates the model which Curses maintains of the prior rendition mode.</p> <p>The <code>vidputs()</code> function computes the terminal output string that <code>vidattr()</code> does, based on <i>attr</i>, but <code>vidputs()</code> outputs by calling the user-supplied function <i>putfunc</i>. The <code>vid_attr()</code> and <code>vid_puts()</code> functions correspond to <code>vidattr()</code> and <code>vidputs()</code> respectively, but take a set of arguments, one of type <code>attr_t</code> for the attributes, one of type <code>short</code> for the color pair number, and a <code>void *</code>, and thus support the attribute constants with the <code>WA_</code> prefix.</p> <p>The <i>opts</i> argument is reserved for definition in a future release. Currently, it is implemented as a null pointer.</p> <p>The user-supplied function <i>putfunc</i> (which can be specified as an argument to either <code>vidputs()</code> or <code>vid_puts()</code>) is either <code>putchar()</code> or some other function with the same prototype. Both the <code>vidputs()</code> and <code>vid_puts()</code> functions ignore the return value of <i>putfunc</i>.</p> | | | | | | | | |
| RETURN VALUES | Upon successful completion, these functions return OK. Otherwise, they return ERR. | | | | | | | | |
| ERRORS | No errors are defined. | | | | | | | | |

vid_attr(3XCURSES)

USAGE After use of any of these functions, the model Curses maintains of the state of the terminal might not match the actual state of the terminal. The application should touch and refresh the window before resuming conventional use of Curses.

Of these functions requires that the application contain so much information about a particular class of terminal that it defeats the purpose of using Curses.

On some terminals, a command to change rendition conceptually occupies space in the screen buffer (with or without width). Thus, a command to set the terminal to a new rendition would change the rendition of some characters already displayed.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO doupdate(3XCURSES), is_linetouched(3XCURSES), libcurses(3XCURSES), putchar(3C), tigetflag(3XCURSES), attributes(5), standards(5)

vidattr(3XCURSES)

| | | | | | | | | | |
|--------------------------|--|-------------|--|--------------------------|------------------|------------|---|----------------|-------------------------------------|
| NAME | vidattr, vid_attr, vidputs, vid_puts – output attributes to the terminal | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int vidattr(chtype attr); int vid_attr(attr_t attr, short color_pair_number, void *opt); int vidputs(chtype attr, int (*putfunc) (int)); int vid_puts(attr_t attr, short color_pair_number, void *opt, int (*putfunc) (int));</pre> | | | | | | | | |
| PARAMETERS | <table><tr><td><i>attr</i></td><td>Is the rendition of the foreground window.</td></tr><tr><td><i>color_pair_number</i></td><td>Is a color pair.</td></tr><tr><td><i>opt</i></td><td>Is reserved for future use. Currently, this must be a null pointer.</td></tr><tr><td><i>putfunc</i></td><td>Is a user-supplied output function.</td></tr></table> | <i>attr</i> | Is the rendition of the foreground window. | <i>color_pair_number</i> | Is a color pair. | <i>opt</i> | Is reserved for future use. Currently, this must be a null pointer. | <i>putfunc</i> | Is a user-supplied output function. |
| <i>attr</i> | Is the rendition of the foreground window. | | | | | | | | |
| <i>color_pair_number</i> | Is a color pair. | | | | | | | | |
| <i>opt</i> | Is reserved for future use. Currently, this must be a null pointer. | | | | | | | | |
| <i>putfunc</i> | Is a user-supplied output function. | | | | | | | | |
| DESCRIPTION | <p>These functions output commands to the terminal that change the terminal's attributes.</p> <p>If the terminfo database indicates that the terminal in use can display characters in the rendition specified by <i>attr</i>, then <code>vidattr()</code> outputs one or more commands to request that the terminal display subsequent characters in that rendition. The function outputs by calling <code>putchar(3C)</code>. The <code>vidattr()</code> function neither relies on your updates the model which Curses maintains of the prior rendition mode.</p> <p>The <code>vidputs()</code> function computes the terminal output string that <code>vidattr()</code> does, based on <i>attr</i>, but <code>vidputs()</code> outputs by calling the user-supplied function <i>putfunc</i>. The <code>vid_attr()</code> and <code>vid_puts()</code> functions correspond to <code>vidattr()</code> and <code>vidputs()</code> respectively, but take a set of arguments, one of type <code>attr_t</code> for the attributes, one of type <code>short</code> for the color pair number, and a <code>void *</code>, and thus support the attribute constants with the <code>WA_</code> prefix.</p> <p>The <i>opts</i> argument is reserved for definition in a future release. Currently, it is implemented as a null pointer.</p> <p>The user-supplied function <i>putfunc</i> (which can be specified as an argument to either <code>vidputs()</code> or <code>vid_puts()</code>) is either <code>putchar()</code> or some other function with the same prototype. Both the <code>vidputs()</code> and <code>vid_puts()</code> functions ignore the return value of <i>putfunc</i>.</p> | | | | | | | | |
| RETURN VALUES | Upon successful completion, these functions return OK. Otherwise, they return ERR. | | | | | | | | |
| ERRORS | No errors are defined. | | | | | | | | |

vidattr(3XCURSES)

USAGE After use of any of these functions, the model Curses maintains of the state of the terminal might not match the actual state of the terminal. The application should touch and refresh the window before resuming conventional use of Curses.

Of these functions requires that the application contain so much information about a particular class of terminal that it defeats the purpose of using Curses.

On some terminals, a command to change rendition conceptually occupies space in the screen buffer (with or without width). Thus, a command to set the terminal to a new rendition would change the rendition of some characters already displayed.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `douupdate(3XCURSES)`, `is_linetouched(3XCURSES)`, `libcurses(3XCURSES)`, `putchar(3C)`, `tigetflag(3XCURSES)`, `attributes(5)`, `standards(5)`

vidputs(3CURSES)

| | |
|--------------------|--|
| NAME | <code> curs_terminfo, setupterm, setterm, set_curterm, del_curterm, restartterm, tparm, tputs, putp, vidputs, vidattr, mvcur, tigetflag, tigetnum, tigetstr</code> – curses interfaces to terminfo database |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> #include <term.h> int setupterm(char *term, int fildes, int *errret); int setterm(char *term); int set_curterm(TERMINAL *nterm); int del_curterm(TERMINAL *oterm); int restartterm(char *term, int fildes, int *errret); char *tparm(char *str, long int p1, long int p2, long int p3, long int p4, long int p5, long int p6, long int p7, long int p8, long int p9); int tputs(char *str, int affcnt, int (*putc)(char)); int putp(char *str); int vidputs(chtype attrs, int (*putc)(char)); int vidattr(chtype attrs); int mvcur(int oldrow, int oldcol, int newrow, int newcol); int tigetflag(char *capname); int tigetnum(char *capname); char *tigetstr(char *capname);</pre> |
| DESCRIPTION | <p>These low-level routines must be called by programs that have to deal directly with the <i>terminfo</i> database to handle certain terminal capabilities, such as programming function keys. For all other functionality, curses routines are more suitable and their use is recommended.</p> <p>Initially, <code>setupterm()</code> should be called. Note that <code>setupterm()</code> is automatically called by <code>initscr()</code> and <code>newterm()</code>. This defines the set of terminal-dependent variables (listed in <code>terminfo(4)</code>). The <i>terminfo</i> variables <code>lines</code> and <code>columns</code> are initialized by <code>setupterm()</code> as follows: If <code>use_env(FALSE)</code> has been called, values for <code>lines</code> and <code>columns</code> specified in <i>terminfo</i> are used. Otherwise, if the environment variables <code>LINES</code> and <code>COLUMNS</code> exist, their values are used. If these environment variables do not exist and the program is running in a window, the current window size is used. Otherwise, if the environment variables do not exist, the values for <code>lines</code> and <code>columns</code> specified in the <i>terminfo</i> database are used.</p> <p>The headers <code><curses.h></code> and <code><term.h></code> should be included (in this order) to get the definitions for these strings, numbers, and flags. Parameterized strings should be passed through <code>tparm()</code> to instantiate them. All <i>terminfo</i> strings (including the output</p> |

of `tparm()` should be printed with `tputs()` or `putp()`. Call the `reset_shell_mode()` routine to restore the tty modes before exiting (see `curs_kernel(3CURSES)`). Programs which use cursor addressing should output `enter_ca_mode` upon startup and should output `exit_ca_mode` before exiting. Programs desiring shell escapes should call `reset_shell_mode` and output `exit_ca_mode` before the shell is called and should output `enter_ca_mode` and call `reset_prog_mode` after returning from the shell.

The `setupterm()` routine reads in the *terminfo* database, initializing the *terminfo* structures, but does not set up the output virtualization structures used by *curses*. The terminal type is the character string *term*; if *term* is null, the environment variable `TERM` is used. All output is to file descriptor *fildev* which is initialized for output. If *errret* is not null, then `setupterm()` returns `OK` or `ERR` and stores a status value in the integer pointed to by *errret*. A status of 1 in *errret* is normal, 0 means that the terminal could not be found, and -1 means that the *terminfo* database could not be found. If *errret* is null, `setupterm()` prints an error message upon finding an error and exits. Thus, the simplest call is:

```
setupterm((char *)0, 1, (int *)0);
```

which uses all the defaults and sends the output to `stdout`.

The `setterm()` routine is being replaced by `setupterm()`. The call:

```
setupterm(term, 1, (int *)0)
```

provides the same functionality as `setterm(term)`. The `setterm()` routine is included here for compatibility and is supported at Level 2.

The `set_curterm()` routine sets the variable `cur_term` to *nterm*, and makes all of the *terminfo* boolean, numeric, and string variables use the values from *nterm*.

The `del_curterm()` routine frees the space pointed to by *oterm* and makes it available for further use. If *oterm* is the same as `cur_term`, references to any of the *terminfo* boolean, numeric, and string variables thereafter may refer to invalid memory locations until another `setupterm()` has been called.

The `restartterm()` routine is similar to `setupterm()` and `initscr()`, except that it is called after restoring memory to a previous state. It assumes that the windows and the input and output options are the same as when memory was saved, but the terminal type and baud rate may be different.

The `tparm()` routine instantiates the string *str* with parameters *pi*. A pointer is returned to the result of *str* with the parameters applied.

The `tputs()` routine applies padding information to the string *str* and outputs it. The *str* must be a *terminfo* string variable or the return value from `tparm()`, `tgetstr()`, or `tgoto()`. *affcnt* is the number of lines affected, or 1 if not applicable. *putc* is a `putchar()`-like routine to which the characters are passed, one at a time.

vidputs(3CURSES)

The `putp()` routine calls `tputs(str, 1, putchar)`. Note that the output of `putpA()` always goes to `stdout`, not to the *files* specified in `setupterm()`.

The `vidputs()` routine displays the string on the terminal in the video attribute mode *attrs*, which is any combination of the attributes listed in `curses(3CURSES)`. The characters are passed to the `putchar()`-like routine `putc()`.

The `vidattr()` routine is like the `vidputs()` routine, except that it outputs through `putchar()`.

The `mvcur()` routine provides low-level cursor motion.

The `tigetflag()`, `tigetnum()` and `tigetstr()` routines return the value of the capability corresponding to the *terminfo capname* passed to them, such as `xenl`.

With the `tigetflag()` routine, the value `-1` is returned if *capname* is not a boolean capability.

With the `tigetnum()` routine, the value `-2` is returned if *capname* is not a numeric capability.

With the `tigetstr()` routine, the value `(char *)-1` is returned if *capname* is not a string capability.

The *capname* for each capability is given in the table column entitled *capname code* in the capabilities section of `terminfo(4)`.

```
char *boolnames, *boolcodes, *boolfnames
char *numnames, *numcodes, *numfnames
char *strnames, *strcodes, *strfnames
```

These null-terminated arrays contain the *capnames*, the *termcap* codes, and the full C names, for each of the *terminfo* variables.

RETURN VALUES

All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

Routines that return pointers always return `NULL` on error.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_initscr(3CURSES)`, `curs_kernel(3CURSES)`, `curs_termcap(3CURSES)`, `curses(3CURSES)`, `putc(3C)`, `terminfo(4)`, `attributes(5)`

vidputs(3CURSES)

NOTES | The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

The `setupterm()` routine should be used in place of `setterm()`.

Note that `vidattr()` and `vidputs()` may be macros.

vid_puts(3XCURSES)

| | | | | | | | | | |
|--------------------------|--|-------------|--|--------------------------|------------------|------------|---|----------------|-------------------------------------|
| NAME | vidattr, vid_attr, vidputs, vid_puts – output attributes to the terminal | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int vidattr(chtype attr); int vid_attr(attr_t attr, short color_pair_number, void *opt); int vidputs(chtype attr, int (*putfunc) (int)); int vid_puts(attr_t attr, short color_pair_number, void *opt, int (*putfunc) (int));</pre> | | | | | | | | |
| PARAMETERS | <table><tr><td><i>attr</i></td><td>Is the rendition of the foreground window.</td></tr><tr><td><i>color_pair_number</i></td><td>Is a color pair.</td></tr><tr><td><i>opt</i></td><td>Is reserved for future use. Currently, this must be a null pointer.</td></tr><tr><td><i>putfunc</i></td><td>Is a user-supplied output function.</td></tr></table> | <i>attr</i> | Is the rendition of the foreground window. | <i>color_pair_number</i> | Is a color pair. | <i>opt</i> | Is reserved for future use. Currently, this must be a null pointer. | <i>putfunc</i> | Is a user-supplied output function. |
| <i>attr</i> | Is the rendition of the foreground window. | | | | | | | | |
| <i>color_pair_number</i> | Is a color pair. | | | | | | | | |
| <i>opt</i> | Is reserved for future use. Currently, this must be a null pointer. | | | | | | | | |
| <i>putfunc</i> | Is a user-supplied output function. | | | | | | | | |
| DESCRIPTION | <p>These functions output commands to the terminal that change the terminal's attributes.</p> <p>If the terminfo database indicates that the terminal in use can display characters in the rendition specified by <i>attr</i>, then <code>vidattr()</code> outputs one or more commands to request that the terminal display subsequent characters in that rendition. The function outputs by calling <code>putchar(3C)</code>. The <code>vidattr()</code> function neither relies on your updates the model which Curses maintains of the prior rendition mode.</p> <p>The <code>vidputs()</code> function computes the terminal output string that <code>vidattr()</code> does, based on <i>attr</i>, but <code>vidputs()</code> outputs by calling the user-supplied function <i>putfunc</i>. The <code>vid_attr()</code> and <code>vid_puts()</code> functions correspond to <code>vidattr()</code> and <code>vidputs()</code> respectively, but take a set of arguments, one of type <code>attr_t</code> for the attributes, one of type <code>short</code> for the color pair number, and a <code>void *</code>, and thus support the attribute constants with the <code>WA_</code> prefix.</p> <p>The <i>opts</i> argument is reserved for definition in a future release. Currently, it is implemented as a null pointer.</p> <p>The user-supplied function <i>putfunc</i> (which can be specified as an argument to either <code>vidputs()</code> or <code>vid_puts()</code>) is either <code>putchar()</code> or some other function with the same prototype. Both the <code>vidputs()</code> and <code>vid_puts()</code> functions ignore the return value of <i>putfunc</i>.</p> | | | | | | | | |
| RETURN VALUES | Upon successful completion, these functions return OK. Otherwise, they return ERR. | | | | | | | | |
| ERRORS | No errors are defined. | | | | | | | | |

vid_puts(3XCURSES)

USAGE After use of any of these functions, the model Curses maintains of the state of the terminal might not match the actual state of the terminal. The application should touch and refresh the window before resuming conventional use of Curses.

Of these functions requires that the application contain so much information about a particular class of terminal that it defeats the purpose of using Curses.

On some terminals, a command to change rendition conceptually occupies space in the screen buffer (with or without width). Thus, a command to set the terminal to a new rendition would change the rendition of some characters already displayed.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `douupdate(3XCURSES)`, `is_linetouched(3XCURSES)`, `libcurses(3XCURSES)`, `putchar(3C)`, `tigetflag(3XCURSES)`, `attributes(5)`, `standards(5)`

vidputs(3XCURSES)

| | | | | | | | | | |
|--------------------------|--|-------------|--|--------------------------|------------------|------------|---|----------------|-------------------------------------|
| NAME | vidattr, vid_attr, vidputs, vid_puts – output attributes to the terminal | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int vidattr(chtype attr); int vid_attr(attr_t attr, short color_pair_number, void *opt); int vidputs(chtype attr, int (*putfunc) (int)); int vid_puts(attr_t attr, short color_pair_number, void *opt, int (*putfunc) (int));</pre> | | | | | | | | |
| PARAMETERS | <table><tr><td><i>attr</i></td><td>Is the rendition of the foreground window.</td></tr><tr><td><i>color_pair_number</i></td><td>Is a color pair.</td></tr><tr><td><i>opt</i></td><td>Is reserved for future use. Currently, this must be a null pointer.</td></tr><tr><td><i>putfunc</i></td><td>Is a user-supplied output function.</td></tr></table> | <i>attr</i> | Is the rendition of the foreground window. | <i>color_pair_number</i> | Is a color pair. | <i>opt</i> | Is reserved for future use. Currently, this must be a null pointer. | <i>putfunc</i> | Is a user-supplied output function. |
| <i>attr</i> | Is the rendition of the foreground window. | | | | | | | | |
| <i>color_pair_number</i> | Is a color pair. | | | | | | | | |
| <i>opt</i> | Is reserved for future use. Currently, this must be a null pointer. | | | | | | | | |
| <i>putfunc</i> | Is a user-supplied output function. | | | | | | | | |
| DESCRIPTION | <p>These functions output commands to the terminal that change the terminal's attributes.</p> <p>If the terminfo database indicates that the terminal in use can display characters in the rendition specified by <i>attr</i>, then <code>vidattr()</code> outputs one or more commands to request that the terminal display subsequent characters in that rendition. The function outputs by calling <code>putchar(3C)</code>. The <code>vidattr()</code> function neither relies on your updates the model which Curses maintains of the prior rendition mode.</p> <p>The <code>vidputs()</code> function computes the terminal output string that <code>vidattr()</code> does, based on <i>attr</i>, but <code>vidputs()</code> outputs by calling the user-supplied function <i>putfunc</i>. The <code>vid_attr()</code> and <code>vid_puts()</code> functions correspond to <code>vidattr()</code> and <code>vidputs()</code> respectively, but take a set of arguments, one of type <code>attr_t</code> for the attributes, one of type <code>short</code> for the color pair number, and a <code>void *</code>, and thus support the attribute constants with the <code>WA_</code> prefix.</p> <p>The <i>opts</i> argument is reserved for definition in a future release. Currently, it is implemented as a null pointer.</p> <p>The user-supplied function <i>putfunc</i> (which can be specified as an argument to either <code>vidputs()</code> or <code>vid_puts()</code>) is either <code>putchar()</code> or some other function with the same prototype. Both the <code>vidputs()</code> and <code>vid_puts()</code> functions ignore the return value of <i>putfunc</i>.</p> | | | | | | | | |
| RETURN VALUES | Upon successful completion, these functions return OK. Otherwise, they return ERR. | | | | | | | | |
| ERRORS | No errors are defined. | | | | | | | | |

vidputs(3XCURSES)

USAGE After use of any of these functions, the model Curses maintains of the state of the terminal might not match the actual state of the terminal. The application should touch and refresh the window before resuming conventional use of Curses.

Of these functions requires that the application contain so much information about a particular class of terminal that it defeats the purpose of using Curses.

On some terminals, a command to change rendition conceptually occupies space in the screen buffer (with or without width). Thus, a command to set the terminal to a new rendition would change the rendition of some characters already displayed.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `douupdate(3XCURSES)`, `is_linetouched(3XCURSES)`, `libcurses(3XCURSES)`, `putchar(3C)`, `tigetflag(3XCURSES)`, `attributes(5)`, `standards(5)`

vline(3XCURSES)

| | |
|----------------------|--|
| NAME | <code>hline</code> , <code>mvhline</code> , <code>mvvline</code> , <code>mvwhline</code> , <code>mvwvline</code> , <code>vline</code> , <code>whline</code> , <code>wvline</code> – use single-byte characters (and renditions) to draw lines |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int hline(chtype ch, int n); int mvhline(int y, int x, chtype ch, int n); int mvvline(int y, int x, chtype ch, int n); int mvwhline(WINDOW *win, int y, int x, chtype ch, int n); int mvwvline(WINDOW *win, int y, int x, chtype ch, int n); int vline(chtype ch, int n); int whline(WINDOW *win, chtype ch, int n); int wvline(WINDOW *win, chtype ch, int n);</pre> |
| DESCRIPTION | <p>The <code>hline()</code>, <code>vline()</code>, <code>whline()</code>, <code>wvline()</code> functions draw a horizontal or vertical line, in either the window <code>stdscr</code> or <code>win</code> starting at the current cursor position. The line is drawn using the character <code>ch</code> and is a maximum of <code>n</code> positions long, or as many as will fit into the window. If <code>ch</code> is 0 (zero), the default horizontal or vertical character is used.</p> <p>The <code>mvhline()</code>, <code>mvvline()</code>, <code>mvwhline()</code>, <code>mvwvline()</code> functions are similar to the previous group of functions but the line begins at cursor position specified by <code>x</code> and <code>y</code>.</p> <p>The functions with names ending with <code>hline()</code> draw horizontal lines proceeding towards the last column of the same line. The functions with names ending with <code>vline()</code> draw vertical lines proceeding towards the last column of the same line.</p> <p>These functions do not change the position of the cursor.</p> |
| PARAMETERS | <p><code>ch</code> Is the character used to draw the line.</p> <p><code>n</code> Is the maximum number of characters in the line.</p> <p><code>y</code> Is the <code>y</code> (row) coordinate for the start of the line.</p> <p><code>x</code> Is the <code>x</code> (column) coordinate for the start of the line.</p> <p><code>win</code> Is a pointer to a window.</p> |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. |
| ERRORS | None |

vline(3XCURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `border(3XCURSES)`, `border_set(3XCURSES)`, `hline_set(3XCURSES)`,
`libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

vline_set(3XCURSES)

| | |
|----------------------|---|
| NAME | <code>hline_set</code> , <code>mvhline_set</code> , <code>mvvline_set</code> , <code>mvwhline_set</code> , <code>mvwvline_set</code> , <code>vline_set</code> , <code>whline_set</code> , <code>wvline_set</code> – use complex characters (and renditions) to draw lines |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int hline_set(const cchar_t *ch, int n); int mvhline_set(int y, int x, const cchar_t *wch, int n); int mvvline_set(int y, int x, const cchar_t *wch, int n); int mvwhline_set(WINDOW *win, int y, int x, const cchar_t *wch, int n); int mvwvline_set(WINDOW *win, int y, int x, const cchar_t *wch, int n); int vline_set(const cchar_t *wch, int n); int whline_set(WINDOW *win, const cchar_t *wch, int n); int wvline_set(WINDOW *win, const cchar_t *wch, int n);</pre> |
| DESCRIPTION | <p>The <code>hline_set()</code>, <code>vline_set()</code>, <code>whline_set()</code>, <code>wvline_set()</code> functions draw a line, in either the window <code>stdscr</code> or <code>win</code> starting at the current cursor position. The line is drawn using the character <code>wch</code> and is a maximum of <code>n</code> positions long, or as many as will fit into the window. If <code>wch</code> is a null pointer, the default horizontal or vertical character is used.</p> <p>The <code>mvhline_set()</code>, <code>mvvline_set()</code>, <code>mvwhline_set()</code>, <code>mvwvline_set()</code> functions are similar to the previous group of functions but the line begins at cursor position specified by <code>x</code> and <code>y</code>.</p> <p>The functions with names ending with <code>hline_set()</code> draw horizontal lines proceeding towards the last column of the same line. The functions with names ending with <code>vline_set()</code> draw vertical lines proceeding towards the last column of the same line.</p> <p>These functions do not change the position of the cursor.</p> |
| PARAMETERS | <p><code>wch</code> Is the complex character used to draw the line.</p> <p><code>n</code> Is the maximum number of characters in the line.</p> <p><code>y</code> Is the y (row) coordinate for the start of the line.</p> <p><code>x</code> Is the x (column) coordinate for the start of the line.</p> <p><code>win</code> Is a pointer to a window.</p> |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. |

vline_set(3XCURSES)

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `border(3XCURSES)`, `border_set(3XCURSES)`, `hline(3XCURSES)`,
`libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

vwprintw(3CURSES)

| NAME | <code> curs_</code> printw, <code>printw</code> , <code>wprintw</code> , <code>mvprintw</code> , <code>mvwprintw</code> , <code>vwprintw</code> – print formatted output in curses windows | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int printw(char *<i>fmt</i>, /* <i>arg</i> */ ...); int wprintw(WINDOW *<i>win</i>, char *<i>fmt</i>, /* <i>arg</i> */ ...); int mvprintw(int <i>y</i>, int <i>x</i>, char *<i>fmt</i>, /* <i>arg</i> */ ...); int mvwprintw(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>, char *<i>fmt</i>, /* <i>arg</i> */...); #include <varargs.h> int vwprintw(WINDOW *<i>win</i>, char *<i>fmt</i>, /* <i>varglist</i> */ ...);</pre> | | | | |
| DESCRIPTION | <p>The <code>printw()</code>, <code>wprintw()</code>, <code>mvprintw()</code>, and <code>mvwprintw()</code> routines are analogous to <code>printf()</code> (see <code>printf(3C)</code>). In effect, the string that would be output by <code>printf()</code> is output instead as though <code>waddstr()</code> were used on the given window.</p> <p>The <code>vwprintw()</code> routine is analogous to <code>vprintf()</code> (see <code>vprintf(3C)</code>) and performs a <code>wprintw()</code> using a variable argument list. The third argument is a <code>va_list</code>, a pointer to a list of arguments, as defined in <code><varargs.h></code>.</p> | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>printf(3C)</code> , <code>vprintf(3C)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . | | | | |

NAME vw_printw – print formatted output in window

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <stdarg.h>
#include <curses.h>

int vw_printw(WINDOW *win, char *fmt, va_list varlist);
```

PARAMETERS

| | |
|----------------|--|
| <i>fmt</i> | Is a printf() format string. |
| <i>varlist</i> | Is a pointer to a list of parameters. |
| <i>win</i> | Is a pointer to the window in which the string is to be written. |

DESCRIPTION The vw_printw() function achieves the same effect as wprintw(3XCURSES) using a variable argument list. The third argument is a va_list, as defined in <stdarg.h>.

RETURN VALUES Upon successful completion, vw_printw() returns OK. Otherwise, it returns ERR.

ERRORS No errors are defined.

USAGE The vw_printw() function is preferred over wprintw(3XCURSES). The use of the wprintw() and vw_printw() in the same file will not work, due to the requirements to include <varargs.h> and <stdarg.h>, which both contain definitions of va_list.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO libcurses(3XCURSES), mvprintw(3XCURSES), printf(3C), attributes(5), standards(5)

vwprintw(3XCURSES)

| NAME | vwprintw – print formatted output in window | | | | | | |
|----------------------|--|----------------|------------------------------|---------------------|---------------------------------------|------------|--|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <varargs.h> #include <curses.h> int vwprintw(WINDOW *win, char *fmt, va_list varlist);</pre> | | | | | | |
| PARAMETERS | <table><tr><td><i>fmt</i></td><td>Is a printf() format string.</td></tr><tr><td><i>varlist</i></td><td>Is a pointer to a list of parameters.</td></tr><tr><td><i>win</i></td><td>Is a pointer to the window in which the string is to be written.</td></tr></table> | <i>fmt</i> | Is a printf() format string. | <i>varlist</i> | Is a pointer to a list of parameters. | <i>win</i> | Is a pointer to the window in which the string is to be written. |
| <i>fmt</i> | Is a printf() format string. | | | | | | |
| <i>varlist</i> | Is a pointer to a list of parameters. | | | | | | |
| <i>win</i> | Is a pointer to the window in which the string is to be written. | | | | | | |
| DESCRIPTION | The vwprintw() function achieves the same effect as wprintw(3XCURSES) using a variable argument list. The third argument is a va_list, as defined in <varargs.h>. | | | | | | |
| RETURN VALUES | Upon successful completion, vwprintw() returns OK. Otherwise, it returns ERR. | | | | | | |
| ERRORS | No errors are defined. | | | | | | |
| USAGE | The vwprintw() function is deprecated; the vw_printw(3XCURSES) function is preferred. The use of the vwprintw() and vw_printw() in the same file will not work, due to the requirements to include <varargs.h> and <stdarg.h>, which both contain definitions of va_list. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Standard</td></tr><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | libcurses(3XCURSES), mvprintw(3XCURSES), printf(3C), vw_printw(3XCURSES), attributes(5), standards(5) | | | | | | |

| NAME | curs_scanw, scanw, wscanw, mvscanw, mvwscanw, vwscanw – convert formatted input from a curses widow | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int scanw(char *fmt, /* arg */ ...); int wscanw(WINDOW *win, char *fmt, /* arg */ ...); int mvscanw(int y, int x, char *fmt, /* arg */ ...); int mvwscanw(WINDOW *win, int y, int x, char *fmt, /* arg */...); int vwscanw(WINDOW *win, char *fmt, va_list varlist);</pre> | | | | |
| DESCRIPTION | <p>The scanw(), wscanw(), and mvscanw() routines correspond to scanf() (see scanf(3C)). The effect of these routines is as though wgetstr() were called on the window, and the resulting line used as input for the scan. Fields which do not map to a variable in the fmt field are lost.</p> <p>The vwscanw() routine is similar to vwprintw() in that it performs a wscanw() using a variable argument list. The third argument is a va_list, a pointer to a list of arguments, as defined in <varargs.h>.</p> | | | | |
| RETURN VALUES | <p>vwscanw() returns ERR on failure and an integer equal to the number of fields scanned on success.</p> <p>Applications may interrogate the return value from the scanw, wscanw(), mvscanw(), and mvwscanw() routines to determine the number of fields which were mapped in the call.</p> | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curs_getstr(3CURSES), curs_printw(3CURSES), curses(3CURSES), scanf(3C), attributes(5) | | | | |
| NOTES | The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>. | | | | |

vw_scanw(3XCURSES)

NAME vw_scanw – convert formatted input from a window

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <stdarg.h>
#include <curses.h>

int vw_scanw(WINDOW *win, char *fmt, va_list varlist);
```

PARAMETERS

| | |
|----------------|--|
| <i>fmt</i> | Is a scanf () format string. |
| <i>varlist</i> | Is a pointer to a list of parameters. |
| <i>win</i> | Is a pointer to the window in which the character is to be read. |

DESCRIPTION The vw_scanw () function achieves the same effect as wscanw(3XCURSES) using a variable argument list. The third argument is a va_list, as defined in <stdarg.h>.

RETURN VALUES Upon successful completion, vw_scanw () returns OK. Otherwise, it returns ERR.

ERRORS No errors are defined.

USAGE The vw_scanw () function is preferred over vwscanw(3XCURSES). The use of the vwscanw () and vw_scanw () in the same file will not work, due to the requirements to include <varargs.h> and <stdarg.h>, which both contain definitions of va_list.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO libcurses(3XCURSES), mvscanw(3XCURSES), scanf(3C), attributes(5), standards(5)

| NAME | vwscanw – convert formatted input from a window | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <varargs.h> #include <curses.h> int vwscanw(WINDOW *win, char *fmt, va_list varlist);</pre> | | | | | | |
| PARAMETERS | <p><i>fmt</i> Is a scanf () format string.</p> <p><i>varlist</i> Is a pointer to a list of parameters.</p> <p><i>win</i> Is a pointer to the window in which the character is to be read.</p> | | | | | | |
| DESCRIPTION | The vwscanw() function achieves the same effect as wscanw(3XCURSES) using a variable argument list. The third argument is a va_list, as defined in <varargs.h>. | | | | | | |
| RETURN VALUES | Upon successful completion, vwscanw() returns OK. Otherwise, it returns ERR. | | | | | | |
| ERRORS | No errors are defined. | | | | | | |
| USAGE | The vwscanw() function is deprecated; the vw_scanw(3XCURSES) function is preferred. The use of the vwscanw() and vw_scanw() in the same file will not work, due to the requirements to include <varargs.h> and <stdarg.h>, which both contain definitions of va_list. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | libcurses(3XCURSES), mvscanw(3XCURSES), scanf(3C), vw_scanw(3XCURSES), attributes(5), standards(5) | | | | | | |

waddch(3CURSES)

| | |
|----------------------|--|
| NAME | <code> curs_addch</code> , <code> addch</code> , <code> waddch</code> , <code> mvaddch</code> , <code> mvwaddch</code> , <code> echochar</code> , <code> wechochar</code> – add a character (with attributes) to a curses window and advance cursor |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int addch(chtype <i>ch</i>); int waddch(WINDOW *<i>win</i>, chtype <i>ch</i>); int mvaddch(int <i>y</i>, int <i>x</i>, chtype <i>ch</i>); int mvwaddch(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>, chtype <i>ch</i>); int echochar(chtype <i>ch</i>); int wechochar(WINDOW *<i>win</i>, chtype <i>ch</i>);</pre> |
| DESCRIPTION | <p>With the <code>addch()</code>, <code>waddch()</code>, <code>mvaddch()</code>, and <code>mvwaddch()</code> routines, the character <i>ch</i> is put into the window at the current cursor position of the window and the position of the window cursor is advanced. Its function is similar to that of <code>putchar()</code>. At the right margin, an automatic newline is performed. At the bottom of the scrolling region, if <code>scrollok()</code> is enabled, the scrolling region is scrolled up one line.</p> <p>If <i>ch</i> is a tab, newline, or backspace, the cursor is moved appropriately within the window. A newline also does a <code>clrtoeol()</code> before moving. Tabs are considered to be at every eighth column. If <i>ch</i> is another control character, it is drawn in the <code>^X</code> notation. Calling <code>winch()</code> after adding a control character does not return the control character, but instead returns the representation of the control character. See <code>curs_inch(3CURSES)</code>.</p> <p>Video attributes can be combined with a character by OR-ing them into the parameter. This results in these attributes also being set. (The intent here is that text, including attributes, can be copied from one place to another using <code>inch()</code> and <code>addch()</code>.) (see <code>standout()</code>, predefined video attribute constants, on the <code>curs_attr(3CURSES)</code> page).</p> <p>The <code>echochar()</code> and <code>wechochar()</code> routines are functionally equivalent to a call to <code>addch()</code> followed by a call to <code>refresh()</code>, or a call to <code>waddch</code> followed by a call to <code>wrefresh()</code>. The knowledge that only a single character is being output is taken into consideration and, for non-control characters, a considerable performance gain might be seen by using these routines instead of their equivalents.</p> |
| Line Graphics | The following variables may be used to add line drawing characters to the screen with routines of the <code>addch()</code> family. When variables are defined for the terminal, the <code>A_ALTCHARSET</code> bit is turned on (see <code>curs_attr(3CURSES)</code>). Otherwise, the default character listed below is stored in the variable. The names chosen are consistent with the VT100 nomenclature. |

waddch(3CURSES)

| Name | Default | Glyph Description |
|--------------|---------|-------------------------|
| ACS_ULCORNER | + | upper left-hand corner |
| ACS_LLCORNER | + | lower left-hand corner |
| ACS_URCORNER | + | upper right-hand corner |
| ACS_LRCORNER | + | lower right-hand corner |
| ACS_RTEE | + | right tee |
| ACS_LTEE | + | left tee |
| ACS_BTEE | + | bottom tee |
| ACS_TTEE | + | top tee |
| ACS_HLINE | - | horizontal line |
| ACS_VLINE | | vertical line |
| ACS_PLUS | + | plus |
| ACS_S1 | - | scan line 1 |
| ACS_S9 | - | scan line 9 |
| ACS_DIAMOND | + | diamond |
| ACS_CKBOARD | : | checker board (stipple) |
| ACS_DEGREE | ' | degree symbol |
| ACS_PLMINUS | # | plus/minus |
| ACS_BULLET | o | bullet |
| ACS_LARROW | < | arrow pointing left |
| ACS_RARROW | > | arrow pointing right |
| ACS_DARROW | v | arrow pointing down |
| ACS_UARROW | ^ | arrow pointing up |
| ACS_BOARD | # | board of squares |
| ACS_LANTERN | # | lantern symbol |
| ACS_BLOCK | # | solid square block |

RETURN VALUES

All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

waddch(3CURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_attr(3CURSES)`, `curl_clear(3CURSES)`, `curl_inch(3CURSES)`, `curl_outopts(3CURSES)`, `curl_refresh(3CURSES)`, `curses(3CURSES)`, `putc(3C)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `addch()`, `mvaddch()`, `mvwaddch()`, and `echochar()` may be macros.

| | | | | | | | | | | | |
|----------------------|---|---------------|---|----------|--|----------|---|----------|--|------------|---|
| NAME | addch, mvaddch, mvwaddch, waddch – add a character (with rendition) to a window | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addch(const chtype ch); int mvaddch(int y, int x, const chtype ch); int mvwaddch(WINDOW *win, int y, int x, const chtype ch); int waddch(WINDOW *win, const chtype ch);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The <code>addch()</code> function writes a character to the <code>stdscr</code> window at the current cursor position. The <code>mvaddch()</code> and <code>mvwaddch()</code> functions write the character to the position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters. The <code>mvaddch()</code> function writes the character to the <code>stdscr</code> window, while <code>mvwaddch()</code> writes the character to the window specified by <code>win</code>. The <code>waddch()</code> function is identical to <code>addch()</code>, but writes the character to the window specified by <code>win</code>.</p> <p>These functions advance the cursor after writing the character. Characters that do not fit on the end of the current line are wrapped to the beginning of the next line unless the current line is the last line of the window and scrolling is disabled. In that situation, characters which extend beyond the end of the line are discarded.</p> <p>When <code>ch</code> is a backspace, carriage return, newline, or tab, X/Open Curses moves the cursor appropriately. Each tab character moves the cursor to the next tab stop. By default, tab stops occur every eight columns. When <code>ch</code> is a control character other than backspace, carriage return, newline, or tab, it is written using <code>^x</code> notation, where <code>x</code> is a printable character. When X/Open Curses writes <code>ch</code> to the last character position on a line, it automatically generates a newline. When <code>ch</code> is written to the last character position of a scrolling region and <code>scrollok()</code> is enabled, X/Open Curses scrolls the scrolling region up one line (see <code>clearok(3XCURSES)</code>).</p> | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 10px;"><i>wchstr</i></td> <td>Is a pointer to the <code>cchar_t</code> string to be copied to the window.</td> </tr> <tr> <td style="padding-right: 10px;"><i>n</i></td> <td>Is the maximum number of characters to be copied from <i>wchstr</i>. If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line.</td> </tr> <tr> <td style="padding-right: 10px;"><i>y</i></td> <td>Is the <code>y</code> (row) coordinate of the starting position of <i>wchstr</i> in the window.</td> </tr> <tr> <td style="padding-right: 10px;"><i>x</i></td> <td>Is the <code>x</code> (column) coordinate of the starting position of <i>wchstr</i> in the window.</td> </tr> <tr> <td style="padding-right: 10px;"><i>win</i></td> <td>Is a pointer to the window to which the string is to be copied.</td> </tr> </table> | <i>wchstr</i> | Is a pointer to the <code>cchar_t</code> string to be copied to the window. | <i>n</i> | Is the maximum number of characters to be copied from <i>wchstr</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | <i>y</i> | Is the <code>y</code> (row) coordinate of the starting position of <i>wchstr</i> in the window. | <i>x</i> | Is the <code>x</code> (column) coordinate of the starting position of <i>wchstr</i> in the window. | <i>win</i> | Is a pointer to the window to which the string is to be copied. |
| <i>wchstr</i> | Is a pointer to the <code>cchar_t</code> string to be copied to the window. | | | | | | | | | | |
| <i>n</i> | Is the maximum number of characters to be copied from <i>wchstr</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | | | | | | | | | | |
| <i>y</i> | Is the <code>y</code> (row) coordinate of the starting position of <i>wchstr</i> in the window. | | | | | | | | | | |
| <i>x</i> | Is the <code>x</code> (column) coordinate of the starting position of <i>wchstr</i> in the window. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window to which the string is to be copied. | | | | | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | | | | | |

waddch(3XCURSES)

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attroff(3XCURSES)`, `bkgdset(3XCURSES)`, `doupage(3XCURSES)`, `inch(3XCURSES)`, `insch(3XCURSES)`, `libcurses(3XCURSES)`, `nl(3XCURSES)`, `printw(3XCURSES)`, `scrollok(3XCURSES)`, `scl(3XCURSES)`, `terminfo(4)`, `attributes(5)`, `standards(5)`

| NAME | curs_addchstr, addchstr, addchnstr, waddchstr, waddchnstr, mvaddchstr, mvaddchnstr, mvwaddchstr, mvwaddchnstr – add string of characters and attributes to a curses window | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int addchstr(chtype *chstr); int addchnstr(chtype *chstr, int n); int waddchstr(WINDOW *win, chtype *chstr); int waddchnstr(WINDOW *win, chtype *chstr, int n); int mvaddchstr(int y, int x, chtype *chstr); int mvaddchnstr(int y, int x, chtype *chstr, int n); int mvwaddchstr(WINDOW *win, int y, int x, chtype *chstr); int mvwaddchnstr(WINDOW *win, int y, int x, chtype *chstr, int n);</pre> | | | | |
| DESCRIPTION | <p>All of these routines copy <i>chstr</i> directly into the window image structure starting at the current cursor position. The four routines with <i>n</i> as the last argument copy at most <i>n</i> elements, but no more than will fit on the line. If <i>n</i>=-1 then the whole string is copied, to the maximum number that fit on the line.</p> <p>The position of the window cursor is not advanced. These routines works faster than <code>waddnstr()</code> (see <code>curs_addstr(3CURSES)</code>) because they merely copy <i>chstr</i> into the window image structure. On the other hand, care must be taken when using these functions because they do not perform any kind of checking (such as for the newline character), they do not advance the current cursor position, and they truncate the string, rather than wrapping it around to the next line.</p> | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_addstr(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that all routines except <code>waddchnstr()</code> and <code>waddchstr()</code> may be macros.</p> | | | | |

waddchnstr(3XCURSES)

| | | | | | | | | | | | |
|----------------------|---|--------------|--|----------|---|----------|--|----------|---|------------|---|
| NAME | addchstr, addchnstr, mvaddchstr, mvaddchnstr, mvwaddchstr, mvwaddchnstr, waddchstr, waddchnstr – copy a character string (with renditions) to a window | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addchstr(const chtype *chstr); int addchnstr(const chtype *chstr, int n); int mvaddchnstr(int y, int x, const chtype *chstr, int n); int mvaddchstr(int y, int x, const chtype *chstr); int mvwaddchnstr(WINDOW *win, int y, int x, const chtype *chstr, int n); int mvwaddchstr(WINDOW *win, int y, int x, const chtype *chstr); int waddchstr(WINDOW *win, const chtype *chstr); int waddchnstr(WINDOW *win, const chtype *chstr, int n);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The <code>addchstr()</code> function copies the <code>chtype</code> character string to the <code>stdscr</code> window at the current cursor position. The <code>mvaddchstr()</code> and <code>mvwaddchstr()</code> functions copy the character string to the starting position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <code>win</code>). The <code>waddchstr()</code> is identical to <code>addchstr()</code>, but writes to the window specified by <code>win</code>.</p> <p>The <code>addchnstr()</code>, <code>waddchnstr()</code>, <code>mvaddchnstr()</code>, and <code>mvwaddchnstr()</code> functions write <code>n</code> characters to the window, or as many as will fit on the line. If <code>n</code> is less than 0, the entire string is written, or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <code>x</code> and <code>y</code> parameters.</p> <p>These functions differ from the <code>addstr(3XCURSES)</code> set of functions in two important respects. First, these functions do <i>not</i> advance the cursor after writing the string to the window. Second, the current window rendition is not combined with the character; only the attributes that are already part of the <code>chtype</code> character are used.</p> | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>chstr</i></td><td>Is a pointer to the <code>chtype</code> string to be copied to the window.</td></tr><tr><td><i>n</i></td><td>Is the maximum number of characters to be copied from <i>chstr</i>. If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line.</td></tr><tr><td><i>y</i></td><td>Is the <code>y</code> (row) coordinate of the starting position of <i>chstr</i> in the window.</td></tr><tr><td><i>x</i></td><td>Is the <code>x</code> (column) coordinate of the starting position of <i>chstr</i> in the window.</td></tr><tr><td><i>win</i></td><td>Is a pointer to the window to which the string is to be copied.</td></tr></table> | <i>chstr</i> | Is a pointer to the <code>chtype</code> string to be copied to the window. | <i>n</i> | Is the maximum number of characters to be copied from <i>chstr</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | <i>y</i> | Is the <code>y</code> (row) coordinate of the starting position of <i>chstr</i> in the window. | <i>x</i> | Is the <code>x</code> (column) coordinate of the starting position of <i>chstr</i> in the window. | <i>win</i> | Is a pointer to the window to which the string is to be copied. |
| <i>chstr</i> | Is a pointer to the <code>chtype</code> string to be copied to the window. | | | | | | | | | | |
| <i>n</i> | Is the maximum number of characters to be copied from <i>chstr</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | | | | | | | | | | |
| <i>y</i> | Is the <code>y</code> (row) coordinate of the starting position of <i>chstr</i> in the window. | | | | | | | | | | |
| <i>x</i> | Is the <code>x</code> (column) coordinate of the starting position of <i>chstr</i> in the window. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window to which the string is to be copied. | | | | | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | | | | | |

waddchnstr(3XCURSES)

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addch(3XCURSES)`, `addnstr(3XCURSES)`, `attroff(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

waddchstr(3CURSES)

| NAME | curs_addchstr, addchstr, addchnstr, waddchstr, waddchnstr, mvaddchstr, mvaddchnstr, mvwaddchstr, mvwaddchnstr – add string of characters and attributes to a curses window | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int addchstr(chtype *chstr); int addchnstr(chtype *chstr, int n); int waddchstr(WINDOW *win, chtype *chstr); int waddchnstr(WINDOW *win, chtype *chstr, int n); int mvaddchstr(int y, int x, chtype *chstr); int mvaddchnstr(int y, int x, chtype *chstr, int n); int mvwaddchstr(WINDOW *win, int y, int x, chtype *chstr); int mvwaddchnstr(WINDOW *win, int y, int x, chtype *chstr, int n);</pre> | | | | |
| DESCRIPTION | <p>All of these routines copy <i>chstr</i> directly into the window image structure starting at the current cursor position. The four routines with <i>n</i> as the last argument copy at most <i>n</i> elements, but no more than will fit on the line. If <i>n</i> = -1 then the whole string is copied, to the maximum number that fit on the line.</p> <p>The position of the window cursor is not advanced. These routines works faster than <code>waddnstr()</code> (see <code>curs_addstr(3CURSES)</code>) because they merely copy <i>chstr</i> into the window image structure. On the other hand, care must be taken when using these functions because they do not perform any kind of checking (such as for the newline character), they do not advance the current cursor position, and they truncate the string, rather than wrapping it around to the next line.</p> | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">MT-Level</td> <td style="text-align: center;">Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_addstr(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that all routines except <code>waddchnstr()</code> and <code>waddchstr()</code> may be macros.</p> | | | | |

| | |
|----------------------|---|
| NAME | addchstr, addchnstr, mvaddchstr, mvaddchnstr, mvwaddchstr, mvwaddchnstr, waddchstr, waddchnstr – copy a character string (with renditions) to a window |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addchstr(const chtype *chstr); int addchnstr(const chtype *chstr, int n); int mvaddchnstr(int y, int x, const chtype *chstr, int n); int mvaddchstr(int y, int x, const chtype *chstr); int mvwaddchnstr(WINDOW *win, int y, int x, const chtype *chstr, int n); int mvwaddchstr(WINDOW *win, int y, int x, const chtype *chstr); int waddchstr(WINDOW *win, const chtype *chstr); int waddchnstr(WINDOW *win, const chtype *chstr, int n);</pre> |
| DESCRIPTION | <p>The <code>addchstr()</code> function copies the <code>chtype</code> character string to the <code>stdscr</code> window at the current cursor position. The <code>mvaddchstr()</code> and <code>mvwaddchstr()</code> functions copy the character string to the starting position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <code>win</code>). The <code>waddchstr()</code> is identical to <code>addchstr()</code>, but writes to the window specified by <code>win</code>.</p> <p>The <code>addchnstr()</code>, <code>waddchnstr()</code>, <code>mvaddchnstr()</code>, and <code>mvwaddchnstr()</code> functions write <code>n</code> characters to the window, or as many as will fit on the line. If <code>n</code> is less than 0, the entire string is written, or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <code>x</code> and <code>y</code> parameters.</p> <p>These functions differ from the <code>addstr(3XCURSES)</code> set of functions in two important respects. First, these functions do <i>not</i> advance the cursor after writing the string to the window. Second, the current window rendition is not combined with the character; only the attributes that are already part of the <code>chtype</code> character are used.</p> |
| PARAMETERS | <p><code>chstr</code> Is a pointer to the <code>chtype</code> string to be copied to the window.</p> <p><code>n</code> Is the maximum number of characters to be copied from <code>chstr</code>. If <code>n</code> is less than 0, the entire string is written or as much of it as fits on the line.</p> <p><code>y</code> Is the <code>y</code> (row) coordinate of the starting position of <code>chstr</code> in the window.</p> <p><code>x</code> Is the <code>x</code> (column) coordinate of the starting position of <code>chstr</code> in the window.</p> <p><code>win</code> Is a pointer to the window to which the string is to be copied.</p> |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. |

waddchstr(3XCURSES)

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addch(3XCURSES)`, `addnstr(3XCURSES)`, `attroff(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_addstr, addstr, addnstr, waddstr, waddnstr, mvaddstr, mvaddnstr, mvwaddstr, mvwaddnstr – add a string of characters to a curses window and advance cursor | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int addstr(char *str); int addnstr(char *str, int n); int waddstr(WINDOW *win, char *str); int waddnstr(WINDOW *win, char *str, int n); int mvaddstr(int y, int x, char *str); int mvaddnstr(int y, int x, char *str, int n); int mvwaddstr(WINDOW *win, int y, int x, char *str); int mvwaddnstr(WINDOW *win, int y, int x, char *str, int n);</pre> | | | | |
| DESCRIPTION | All of these routines write all the characters of the null terminated character string <i>str</i> on the given window. It is similar to calling <code>waddch()</code> once for each character in the string. The four routines with <i>n</i> as the last argument write at most <i>n</i> characters. If <i>n</i> is negative, then the entire string will be added. | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_addch(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that all routines except <code>waddstr()</code> and <code>waddnstr()</code> may not be macros.</p> | | | | |

waddnstr(3XCURSES)

| | | | | | | | |
|--------------------|---|------------|---|----------|---|----------|--|
| NAME | <code>addnstr</code> , <code>addstr</code> , <code>mvaddnstr</code> , <code>mvaddstr</code> , <code>mvwaddnstr</code> , <code>mvwaddstr</code> , <code>waddnstr</code> , <code>waddstr</code> – add a multi-byte character string (without rendition) to a window | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addnstr(const char *str, int n); int addstr(const char *str); int mvaddnstr(int y, int x, const char *str, int n); int mvaddstr(int y, int x, const char *str); int mvwaddnstr(WINDOW *win, int y, int x, const char *str, int n); int mvwaddstr(WINDOW *win, int y, int x, const char *str); int waddstr(WINDOW *win, const char *str); int waddnstr(WINDOW *win, const char *str, int n);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>addstr()</code> function writes a null-terminated string of multi-byte characters to the <code>stdscr</code> window at the current cursor position. The <code>waddstr()</code> function performs an identical action, but writes the character to the window specified by <i>win</i>. The <code>mvaddstr()</code> and <code>mvwaddstr()</code> functions write the string to the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <i>win</i>).</p> <p>The <code>addnstr()</code>, <code>waddnstr()</code>, <code>mvaddnstr()</code>, and <code>mvwaddnstr()</code> functions are similar but write at most <i>n</i> characters to the window. If <i>n</i> is less than 0, the entire string is written.</p> <p>All of these functions advance the cursor after writing the string.</p> <p>These functions are functionally equivalent to calling the corresponding function from the <code>addch(3XCURSES)</code> set of functions once for each character in the string. Refer to the <code>curses(3XCURSES)</code> man page for a complete description of special character handling and of the interaction between the window rendition (or background character and rendition) and the character written.</p> <p>Note that these functions differ from the <code>addchstr()</code> set of functions in that the <code>addchstr(3XCURSES)</code> functions copy the string as is (without combining each character with the window rendition or the background character and rendition).</p> | | | | | | |
| PARAMETERS | <table><tr><td><i>str</i></td><td>Is a pointer to the character string that is to be written to the window.</td></tr><tr><td><i>n</i></td><td>Is the maximum number of characters to be copied from <i>str</i>. If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line.</td></tr><tr><td><i>y</i></td><td>Is the <i>y</i> (row) coordinate of the starting position of <i>str</i> in the window.</td></tr></table> | <i>str</i> | Is a pointer to the character string that is to be written to the window. | <i>n</i> | Is the maximum number of characters to be copied from <i>str</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of <i>str</i> in the window. |
| <i>str</i> | Is a pointer to the character string that is to be written to the window. | | | | | | |
| <i>n</i> | Is the maximum number of characters to be copied from <i>str</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | | | | | | |
| <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of <i>str</i> in the window. | | | | | | |

waddnstr(3XCURSES)

x Is the x (column) coordinate of the starting position of *str* in the window.

win Is a pointer to the window in which the string is to be written.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addch(3XCURSES)`, `addchstr(3XCURSES)`, `curses(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

waddnwstr(3CURSES)

| NAME | <code> curs_ addwstr, addwstr, addnwstr, waddwstr, waddnwstr, mvaddwstr, mvaddnwstr, mvwaddwstr, mvwaddnwstr </code> – add a string of <code>wchar_t</code> characters to a curses window and advance cursor | | | | |
|---------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lcurses [library...] #include<curses.h> int addwstr(wchar_t *wstr); int addnwstr(wchar_t *wstr, int n); int waddwstr(WINDOW *win, wchar_t *wstr); int waddnwstr(WINDOW *win, wchar_t *wstr, int n); int mvaddwstr(int y, int x, wchar_t *wstr); int mvaddnwstr(int y, int x, wchar_t *wstr, int n); int mvwaddwstr(WINDOW *win, int y, int x, wchar_t *wstr); int mvwaddnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);</pre> | | | | |
| DESCRIPTION | All of these routines write all the characters of the null-terminated <code>wchar_t</code> character string <code>wstr</code> on the given window. The effect is similar to calling <code>waddwch(3CURSES)</code> once for each <code>wchar_t</code> character in the string. The four routines with <code>n</code> as the last argument write at most <code>n</code> <code>wchar_t</code> characters. If <code>n</code> is negative, then the entire string will be added. | | | | |
| RETURN VALUE | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>waddwch(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header file <code><curses.h></code> automatically includes the header files <code><stdio.h></code> , <code><nctrl.h></code> and <code><widec.h></code> . Note that all of these routines except <code>waddwstr()</code> and <code>waddnwstr()</code> may be macros. | | | | |

| | |
|--------------------|---|
| NAME | addnwstr, addwstr, mvaddnwstr, mvaddwstr, mvwaddnwstr, mvwaddwstr, waddnwstr, waddwstr – add a wide-character string to a window |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addnwstr(const wchar_t *wstr, int n); int addwstr(const wchar_t *wstr); int mvaddnwstr(int y, int x, const wchar_t *wstr, int n); int mvaddwstr(int y, int x, const wchar_t *wstr); int mvwaddnwstr(WINDOW*win, int y, int x, const wchar_t *wstr, int n); int mvwaddwstr(WINDOW*win, int y, int x, const wchar_t *wstr); int waddnwstr(WINDOW*win, const wchar_t *wstr, int n); int waddwstr(WINDOW*win, const wchar_t *wstr);</pre> |
| DESCRIPTION | <p>The <code>addwstr()</code> function writes a null-terminated wide-character string to the <code>stdscr</code> window at the current cursor position. The <code>waddwstr()</code> function performs an identical action, but writes the string to the window specified by <code>win</code>. The <code>mvaddwstr()</code> and <code>mvwaddwstr()</code> functions write the string to the position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <code>win</code>).</p> <p>The <code>addnwstr()</code>, <code>waddnwstr()</code>, <code>mvaddnwstr()</code>, and <code>mvwaddnwstr()</code> functions write at most <code>n</code> characters to the window. If <code>n</code> is less than 0, the entire string is written. The former two functions place the characters at the current cursor position; the latter two commands use the position specified by the <code>x</code> and <code>y</code> parameters.</p> <p>All of these functions advance the cursor after writing the string.</p> <p>These functions are functionally equivalent to building a <code>cchar_t</code> from the <code>wchar_t</code> and the window rendition (or background character and rendition) and calling the <code>wadd_wch(3XCURSES)</code> function once for each <code>wchar_t</code> in the string. Refer to the <code>curses(3XCURSES)</code> man page for a complete description of special character handling and of the interaction between the window rendition (or background character and rendition) and the character written.</p> <p>Note that these functions differ from the <code>add_wchnstr(3XCURSES)</code> set of functions in that the latter copy the string as is (without combining each character with the foreground and background attributes of the window).</p> |
| PARAMETERS | <p><code>wstr</code> Is a pointer to the wide-character string that is to be written to the window.</p> |

waddnwstr(3XCURSES)

- n* Is the maximum number of characters to be copied from *wstr*. If *n* is less than 0, the entire string is written or as much of it as fits on the line.
- y* Is the y (row) coordinate of the starting position of *wstr* in the window.
- x* Is the x (column) coordinate of the starting position of *wstr* in the window.
- win* Is a pointer to the window in which the string is to be written.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `add_wch(3XCURSES)`, `add_wchnstr(3XCURSES)`, `curses(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_addstr, addstr, addnstr, waddstr, waddnstr, mvaddstr, mvaddnstr, mvwaddstr, mvwaddnstr – add a string of characters to a curses window and advance cursor | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int addstr(char *str); int addnstr(char *str, int n); int waddstr(WINDOW *win, char *str); int waddnstr(WINDOW *win, char *str, int n); int mvaddstr(int y, int x, char *str); int mvaddnstr(int y, int x, char *str, int n); int mvwaddstr(WINDOW *win, int y, int x, char *str); int mvwaddnstr(WINDOW *win, int y, int x, char *str, int n);</pre> | | | | |
| DESCRIPTION | All of these routines write all the characters of the null terminated character string <i>str</i> on the given window. It is similar to calling <code>waddch()</code> once for each character in the string. The four routines with <i>n</i> as the last argument write at most <i>n</i> characters. If <i>n</i> is negative, then the entire string will be added. | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_addch(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that all routines except <code>waddstr()</code> and <code>waddnstr()</code> may not be macros.</p> | | | | |

waddstr(3XCURSES)

| | | | | | | | |
|--------------------|---|------------|---|----------|---|----------|--|
| NAME | <code>addnstr</code> , <code>addstr</code> , <code>mvaddnstr</code> , <code>mvaddstr</code> , <code>mvwaddnstr</code> , <code>mvwaddstr</code> , <code>waddnstr</code> , <code>waddstr</code> – add a multi-byte character string (without rendition) to a window | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addnstr(const char *str, int n); int addstr(const char *str); int mvaddnstr(int y, int x, const char *str, int n); int mvaddstr(int y, int x, const char *str); int mvwaddnstr(WINDOW *win, int y, int x, const char *str, int n); int mvwaddstr(WINDOW *win, int y, int x, const char *str); int waddstr(WINDOW *win, const char *str); int waddnstr(WINDOW *win, const char *str, int n);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>addstr()</code> function writes a null-terminated string of multi-byte characters to the <code>stdscr</code> window at the current cursor position. The <code>waddstr()</code> function performs an identical action, but writes the character to the window specified by <i>win</i>. The <code>mvaddstr()</code> and <code>mvwaddstr()</code> functions write the string to the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <i>win</i>).</p> <p>The <code>addnstr()</code>, <code>waddnstr()</code>, <code>mvaddnstr()</code>, and <code>mvwaddnstr()</code> functions are similar but write at most <i>n</i> characters to the window. If <i>n</i> is less than 0, the entire string is written.</p> <p>All of these functions advance the cursor after writing the string.</p> <p>These functions are functionally equivalent to calling the corresponding function from the <code>addch(3XCURSES)</code> set of functions once for each character in the string. Refer to the <code>curses(3XCURSES)</code> man page for a complete description of special character handling and of the interaction between the window rendition (or background character and rendition) and the character written.</p> <p>Note that these functions differ from the <code>addchstr()</code> set of functions in that the <code>addchstr(3XCURSES)</code> functions copy the string as is (without combining each character with the window rendition or the background character and rendition).</p> | | | | | | |
| PARAMETERS | <table><tr><td><i>str</i></td><td>Is a pointer to the character string that is to be written to the window.</td></tr><tr><td><i>n</i></td><td>Is the maximum number of characters to be copied from <i>str</i>. If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line.</td></tr><tr><td><i>y</i></td><td>Is the <i>y</i> (row) coordinate of the starting position of <i>str</i> in the window.</td></tr></table> | <i>str</i> | Is a pointer to the character string that is to be written to the window. | <i>n</i> | Is the maximum number of characters to be copied from <i>str</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of <i>str</i> in the window. |
| <i>str</i> | Is a pointer to the character string that is to be written to the window. | | | | | | |
| <i>n</i> | Is the maximum number of characters to be copied from <i>str</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | | | | | | |
| <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of <i>str</i> in the window. | | | | | | |

waddstr(3XCURSES)

x Is the x (column) coordinate of the starting position of *str* in the window.

win Is a pointer to the window in which the string is to be written.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addch(3XCURSES)`, `addchstr(3XCURSES)`, `curses(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

waddwch(3CURSES)

| | |
|----------------------|--|
| NAME | <code> curs_addwch, addwch, waddwch, mvaddwch, mvwaddwch, echowchar, wechowchar</code> – add a <code>wchar_t</code> character (with attributes) to a curses window and advance cursor |
| SYNOPSIS | <pre>cc [flag...] file... -lcurses [library...] #include<curses.h> int addwch(chtype <i>wch</i>); int waddwch(WINDOW *<i>win</i>, chtype <i>wch</i>); int mvaddwch(int <i>y</i>, int <i>x</i>, chtype <i>wch</i>); int mvwaddwch(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>, chtype <i>wch</i>); int echowchar(chtype <i>wch</i>); int wechowchar(WINDOW *<i>win</i>, chtype <i>wch</i>);</pre> |
| DESCRIPTION | <p>The <code>addwch()</code>, <code>waddwch()</code>, <code>mvaddwch()</code>, and <code>mvwaddwch()</code> routines put the character <code>wch</code>, holding a <code>wchar_t</code> character, into the window at the current cursor position of the window and advance the position of the window cursor. Their function is similar to that of <code>putwchar(3C)</code> in the C multibyte library. At the right margin, an automatic newline is performed. At the bottom of the scrolling region, if <code>scrollok</code> is enabled, the scrolling region is scrolled up one line.</p> <p>If <code>wch</code> is a tab, newline, or backspace, the cursor is moved appropriately within the window. A newline also does a <code>clrtoeol(3CURSES)</code> before moving. Tabs are considered to be at every eighth column. If <code>wch</code> is another control character, it is drawn in the <code>^X</code> notation. Calling <code>winwch(3CURSES)</code> after adding a control character does not return the control character, but instead returns the representation of the control character.</p> <p>Video attributes can be combined with a <code>wchar_t</code> character by OR-ing them into the parameter. This results in these attributes also being set. (The intent here is that text, including attributes, can be copied from one place to another using <code>inwch()</code> and <code>addwch()</code>.) See <code>standout(3CURSES)</code>, predefined video attribute constants.</p> <p>The <code>echowchar()</code> and <code>wechowchar()</code> routines are functionally equivalent to a call to <code>addwch()</code> followed by a call to <code>refresh(3CURSES)</code>, or a call to <code>waddwch()</code> followed by a call to <code>wrefresh(3CURSES)</code>. The knowledge that only a single character is being output is taken into consideration and, for non-control characters, a considerable performance gain might be seen by using these routines instead of their equivalents.</p> |
| Line Graphics | <p>The following variables may be used to add line drawing characters to the screen with routines of the <code>addwch()</code> family. When variables are defined for the terminal, the <code>A_ALTCHARSET</code> bit is turned on. (See <code> curs_attr(3CURSES)</code>). Otherwise, the default character listed below is stored in the variable. The names chosen are consistent with the VT100 nomenclature.</p> |

waddwch(3CURSES)

| Name | Default | Glyph Description |
|--------------|---------|-------------------------|
| ACS_ULCORNER | + | upper left-hand corner |
| ACS_LLCORNER | + | lower left-hand corner |
| ACS_URCORNER | + | upper right-hand corner |
| ACS_LRCORNER | + | lower right-hand corner |
| ACS_RTEE | + | right tee |
| ACS_LTEE | + | left tee |
| ACS_BTEE | + | bottom tee |
| ACS_TTEE | + | top tee |
| ACS_HLINE | - | horizontal line |
| ACS_VLINE | | vertical line |
| ACS_PLUS | + | plus |
| ACS_S1 | - | scan line 1 |
| ACS_S9 | - | scan line 9 |
| ACS_DIAMOND | + | diamond |
| ACS_CKBOARD | : | checker board (stipple) |
| ACS_DEGREE | ' | degree symbol |
| ACS_PLMINUS | # | plus/minus |
| ACS_BULLET | o | bullet |
| ACS_LARROW | < | arrow pointing left |
| ACS_RARRROW | > | arrow pointing right |
| ACS_DARROW | v | arrow pointing down |
| ACS_UARROW | ^ | arrow pointing up |
| ACS_BOARD | # | board of squares |
| ACS_LANTERN | # | lantern symbol |
| ACS_BLOCK | # | solid square block |

RETURN VALUE

All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion, unless otherwise noted in the preceding routine descriptions.

waddwch(3CURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `putwchar(3C)`, `clrtoeol(3CURSES)`, `curses(3CURSES)`, `curl_attr(3CURSES)`, `curl_inwch(3CURSES)`, `curl_outopts(3CURSES)`, `refresh(3CURSES)`, `standout(3CURSES)`, `winwch(3CURSES)`, `wrefresh(3CURSES)`, `attributes(5)`

NOTES The header file `<curses.h>` automatically includes the header files `<stdio.h>`, `<unctrl.h>` and `<widec.h>`.

Note that `addwch()`, `mvaddwch()`, `mvwaddwch()`, and `echowchar()` may be macros.

None of these routines can use the color attribute in `chtype`.

| | |
|--------------------|--|
| NAME | add_wch, mvadd_wch, mvwadd_wch, wadd_wch – add a complex character (with rendition) to a window |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int add_wch(const cchar_t *wch); int wadd_wch(WINDOW *win, const cchar_t *wch); int mvadd_wch(int y, int x, const cchar_t *wch); int mvwadd_wch(WINDOW *win, int y, int x, const cchar_t *wch);</pre> |
| DESCRIPTION | <p>The <code>add_wch()</code> function writes a complex character to the <code>stdscr</code> window at the current cursor position. The <code>mvadd_wch()</code> and <code>mvwadd_wch()</code> functions write the character to the position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters. The <code>mvadd_wch()</code> function writes the character to the <code>stdscr</code> window, while <code>mvwadd_wch()</code> writes the character to the window specified by <code>win</code>. The <code>wadd_wch()</code> function is identical to <code>add_wch()</code>, but writes the character to the window specified by <code>win</code>. These functions advance the cursor after writing the character.</p> <p>If <code>wch</code> is a spacing complex character, X/Open Curses replaces any previous character at the specified location with <code>wch</code> (and its rendition). If <code>wch</code> is a non-spacing complex character, X/Open Curses preserves all existing characters at the specified location and adds the non-spacing characters of <code>wch</code> to the spacing complex character. It ignores the rendition associated with <code>wch</code>.</p> <p>Characters that do not fit on the end of the current line are wrapped to the beginning of the next line unless the current line is the last line of the window and scrolling is disabled. In that situation, X/Open Curses discards characters which extend beyond the end of the line.</p> <p>When <code>wch</code> is a backspace, carriage return, newline, or tab, X/Open Curses moves the cursor appropriately as described in the <code>curses(3XCURSES)</code> man page. Each tab character moves the cursor to the next tab stop. By default, tab stops occur every eight columns. When <code>wch</code> is a control character other than a backspace, carriage return, newline, or tab, it is written using <code>^x</code> notation, where <code>x</code> is a printable character. When X/Open Curses writes <code>wch</code> to the last character position on a line, it automatically generates a newline. When <code>wch</code> is written to the last character position of a scrolling region and <code>scrollok()</code> is enabled, X/Open Curses scrolls the scrolling region up one line (see <code>clearok(3XCURSES)</code>).</p> |
| PARAMETERS | <p><code>wch</code> Is the character/attribute pair (rendition) to be written to the window.</p> <p><code>win</code> Is a pointer to the window in which the character is to be written.</p> <p><code>y</code> Is the <code>y</code> (row) coordinate of the character's position in the window.</p> |

wadd_wch(3XCURSES)

x Is the *x* (column) coordinate of the character's position in the window.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `attr_off(3XCURSES)`, `bkgrndset(3XCURSES)`, `curses(3XCURSES)`, `doupdate(3XCURSES)`, `in_wch(3XCURSES)`, `ins_wch(3XCURSES)`, `libcurses(3XCURSES)`, `n1(3XCURSES)`, `printw(3XCURSES)`, `scrollok(3XCURSES)`, `scl1(3XCURSES)`, `setscrreg(3XCURSES)`, `terminfo(4)`, `attributes(5)`, `standards(5)`

| NAME | curs_addwchstr, addwchstr, addwchnstr, waddwchstr, waddwchnstr, mvaddwchstr, mvaddwchnstr, mvwaddwchstr, mvwaddwchnstr – add string of wchar_t characters (and attributes) to a curses window | | | | |
|---------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lcurses [library...] #include<curses.h> int addwchstr (chtype *wchstr) ; int addwchnstr (chtype *wchstr, int n) ; int waddwchstr (WINDOW *win, chtype *wchstr) ; int waddwchnstr (WINDOW *win, chtype *wchstr, int n) ; int mvaddwchstr (int y, int x, chtype *wchstr) ; int mvaddwchnstr (int y, int x, chtype *wchstr, int n) ; int mvwaddwchstr (WINDOW *win, int y, int x, chtype *wchstr) ; int mvwaddwchnstr (WINDOW *win, int y, int x, chtype *wchstr, int n) ;</pre> | | | | |
| DESCRIPTION | <p>All of these routines copy <i>wchstr</i>, which points to a string of wchar_t characters, directly into the window image structure starting at the current cursor position. The four routines with <i>n</i> as the last argument copy at most <i>n</i> elements, but no more than will fit on the line. If <i>n</i>=-1 then the whole string is copied, to the maximum number that fit on the line.</p> <p>The position of the window cursor is not advanced. These routines work faster than waddnwstr(3CURSES) because they merely copy <i>wchstr</i> into the window image structure. On the other hand, care must be taken when using these functions because they don't perform any kind of checking (such as for the newline character), they do not advance the current cursor position, and they truncate the string, rather than wrapping it around to the new line.</p> | | | | |
| RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion, unless otherwise noted in the preceding routine descriptions. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), waddnwstr(3CURSES), attributes(5) | | | | |
| NOTES | The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <widec.h>. | | | | |

waddwchnstr(3CURSES)

Note that all routines except `waddwchnstr()` may be macros.

None of these routines can use the color attribute in `chtype`.

| | | | | | | | |
|--------------------|--|---------------|--|----------|--|----------|--|
| NAME | add_wchnstr, add_wchstr, mvadd_wchnstr, mvadd_wchstr, mvwadd_wchnstr, mvwadd_wchstr, wadd_wchnstr, wadd_wchstr – copy a string of complex characters (with renditions) to a window | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int add_wchnstr(const cchar_t *wchstr, int n); int add_wchstr(const cchar_t *wchstr); int mvadd_wchnstr(int y, int x, const cchar_t *wchstr, int n); int mvadd_wchstr(int y, int x, const cchar_t *wchstr); int mvwadd_wchnstr(WINDOW *win, int y, int x, const cchar_t *wchstr, int n); int mvwadd_wchstr(WINDOW *win, int y, int x, const cchar_t *wchstr); int wadd_wchstr(WINDOW *win, const cchar_t *wchstr); int wadd_wchnstr(WINDOW *win, const cchar_t *wchstr, int n);</pre> | | | | | | |
| DESCRIPTION | <p>The add_wchstr() function copies the string of cchar_t characters to the stdscr window at the current cursor position. The mvadd_wchstr() and mvwadd_wchstr() functions copy the string to the starting position indicated by the x (column) and y (row) parameters (the former to the stdscr window; the latter to window win). The wadd_wchstr() is identical to add_wchstr(), but writes to the window specified by win.</p> <p>The add_wchnstr(), wadd_wchnstr(), mvadd_wchnstr(), and mvwadd_wchnstr() functions write n characters to the window, or as many as will fit on the line. If n is less than 0, the entire string is written, or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the x and y parameters.</p> <p>These functions differ from the addwstr(3XCURSES) set of functions in two important respects. First, these functions do <i>not</i> advance the cursor after writing the string to the window. Second, the current window rendition (that is, the combination of attributes and color pair) is not combined with the character; only those attributes that are already part of the cchar_t character are used.</p> | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>wchstr</i></td> <td>Is a pointer to the cchar_t string to be copied to the window.</td> </tr> <tr> <td><i>n</i></td> <td>Is the maximum number of characters to be copied from <i>wchstr</i>. If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line.</td> </tr> <tr> <td><i>y</i></td> <td>Is the y (row) coordinate of the starting position of <i>wchstr</i> in the window.</td> </tr> </table> | <i>wchstr</i> | Is a pointer to the cchar_t string to be copied to the window. | <i>n</i> | Is the maximum number of characters to be copied from <i>wchstr</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | <i>y</i> | Is the y (row) coordinate of the starting position of <i>wchstr</i> in the window. |
| <i>wchstr</i> | Is a pointer to the cchar_t string to be copied to the window. | | | | | | |
| <i>n</i> | Is the maximum number of characters to be copied from <i>wchstr</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | | | | | | |
| <i>y</i> | Is the y (row) coordinate of the starting position of <i>wchstr</i> in the window. | | | | | | |

wadd_wchnstr(3XCURSES)

x Is the x (column) coordinate of the starting position of *wchstr* in the window.

win Is a pointer to the window to which the string is to be copied.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addnwstr(3XCURSES)`, `add_wch(3XCURSES)`, `attr_off(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_addwchstr, addwchstr, addwchnstr, waddwchstr, waddwchnstr, mvaddwchstr, mvaddwchnstr, mvwaddwchstr, mvwaddwchnstr – add string of wchar_t characters (and attributes) to a curses window | | | | |
|---------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lcurses [library...] #include<curses.h> int addwchstr (chtype *wchstr) ; int addwchnstr (chtype *wchstr, int n) ; int waddwchstr (WINDOW *win, chtype *wchstr) ; int waddwchnstr (WINDOW *win, chtype *wchstr, int n) ; int mvaddwchstr (int y, int x, chtype *wchstr) ; int mvaddwchnstr (int y, int x, chtype *wchstr, int n) ; int mvwaddwchstr (WINDOW *win, int y, int x, chtype *wchstr) ; int mvwaddwchnstr (WINDOW *win, int y, int x, chtype *wchstr, int n) ;</pre> | | | | |
| DESCRIPTION | <p>All of these routines copy <i>wchstr</i>, which points to a string of wchar_t characters, directly into the window image structure starting at the current cursor position. The four routines with <i>n</i> as the last argument copy at most <i>n</i> elements, but no more than will fit on the line. If <i>n</i>==-1 then the whole string is copied, to the maximum number that fit on the line.</p> <p>The position of the window cursor is not advanced. These routines work faster than waddnwstr(3CURSES) because they merely copy <i>wchstr</i> into the window image structure. On the other hand, care must be taken when using these functions because they don't perform any kind of checking (such as for the newline character), they do not advance the current cursor position, and they truncate the string, rather than wrapping it around to the new line.</p> | | | | |
| RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion, unless otherwise noted in the preceding routine descriptions. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), waddnwstr(3CURSES), attributes(5) | | | | |
| NOTES | The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <widec.h>. | | | | |

waddwchstr(3CURSES)

Note that all routines except `waddwchnstr()` may be macros.

None of these routines can use the color attribute in `chtype`.

| | | | | | | | |
|--------------------|--|---------------|---|----------|--|----------|---|
| NAME | add_wchnstr, add_wchstr, mvadd_wchnstr, mvadd_wchstr, mvwadd_wchnstr, mvwadd_wchstr, wadd_wchnstr, wadd_wchstr – copy a string of complex characters (with renditions) to a window | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int add_wchnstr(const cchar_t *wchstr, int n); int add_wchstr(const cchar_t *wchstr); int mvadd_wchnstr(int y, int x, const cchar_t *wchstr, int n); int mvadd_wchstr(int y, int x, const cchar_t *wchstr); int mvwadd_wchnstr(WINDOW *win, int y, int x, const cchar_t *wchstr, int n); int mvwadd_wchstr(WINDOW *win, int y, int x, const cchar_t *wchstr); int wadd_wchstr(WINDOW *win, const cchar_t *wchstr); int wadd_wchnstr(WINDOW *win, const cchar_t *wchstr, int n);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>add_wchstr()</code> function copies the string of <code>cchar_t</code> characters to the <code>stdscr</code> window at the current cursor position. The <code>mvadd_wchstr()</code> and <code>mvwadd_wchstr()</code> functions copy the string to the starting position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <code>win</code>). The <code>wadd_wchstr()</code> is identical to <code>add_wchstr()</code>, but writes to the window specified by <code>win</code>.</p> <p>The <code>add_wchnstr()</code>, <code>wadd_wchnstr()</code>, <code>mvadd_wchnstr()</code>, and <code>mvwadd_wchnstr()</code> functions write <code>n</code> characters to the window, or as many as will fit on the line. If <code>n</code> is less than 0, the entire string is written, or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <code>x</code> and <code>y</code> parameters.</p> <p>These functions differ from the <code>addwstr(3XCURSES)</code> set of functions in two important respects. First, these functions do <i>not</i> advance the cursor after writing the string to the window. Second, the current window rendition (that is, the combination of attributes and color pair) is not combined with the character; only those attributes that are already part of the <code>cchar_t</code> character are used.</p> | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>wchstr</i></td> <td>Is a pointer to the <code>cchar_t</code> string to be copied to the window.</td> </tr> <tr> <td><i>n</i></td> <td>Is the maximum number of characters to be copied from <i>wchstr</i>. If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line.</td> </tr> <tr> <td><i>y</i></td> <td>Is the <i>y</i> (row) coordinate of the starting position of <i>wchstr</i> in the window.</td> </tr> </table> | <i>wchstr</i> | Is a pointer to the <code>cchar_t</code> string to be copied to the window. | <i>n</i> | Is the maximum number of characters to be copied from <i>wchstr</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of <i>wchstr</i> in the window. |
| <i>wchstr</i> | Is a pointer to the <code>cchar_t</code> string to be copied to the window. | | | | | | |
| <i>n</i> | Is the maximum number of characters to be copied from <i>wchstr</i> . If <i>n</i> is less than 0, the entire string is written or as much of it as fits on the line. | | | | | | |
| <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of <i>wchstr</i> in the window. | | | | | | |

wadd_wchstr(3XCURSES)

x Is the x (column) coordinate of the starting position of *wchstr* in the window.
win Is a pointer to the window to which the string is to be copied.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addnwstr(3XCURSES)`, `add_wch(3XCURSES)`, `attr_off(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_addwstr, addwstr, addnwstr, waddwstr, waddnwstr, mvaddwstr, mvaddnwstr, mvwaddwstr, mvwaddnwstr – add a string of wchar_t characters to a curses window and advance cursor | | | | |
|---------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -lcurses [library...] #include<curses.h> int addwstr(wchar_t *wstr); int addnwstr(wchar_t *wstr, int n); int waddwstr(WINDOW *win, wchar_t *wstr); int waddnwstr(WINDOW *win, wchar_t *wstr, int n); int mvaddwstr(int y, int x, wchar_t *wstr); int mvaddnwstr(int y, int x, wchar_t *wstr, int n); int mvwaddwstr(WINDOW *win, int y, int x, wchar_t *wstr); int mvwaddnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);</pre> | | | | |
| DESCRIPTION | All of these routines write all the characters of the null-terminated wchar_t character string wstr on the given window. The effect is similar to calling waddwch(3CURSES) once for each wchar_t character in the string. The four routines with n as the last argument write at most n wchar_t characters. If n is negative, then the entire string will be added. | | | | |
| RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), waddwch(3CURSES), attributes(5) | | | | |
| NOTES | <p>The header file <curses.h> automatically includes the header files <stdio.h>, <nctrl.h> and <widec.h>.</p> <p>Note that all of these routines except waddwstr() and waddnwstr() may be macros.</p> | | | | |

waddwstr(3XCURSES)

| | |
|--------------------|---|
| NAME | <code>addnwstr</code> , <code>addwstr</code> , <code>mvaddnwstr</code> , <code>mvaddwstr</code> , <code>mvwaddnwstr</code> , <code>mvwaddwstr</code> , <code>waddnwstr</code> , <code>waddwstr</code> – add a wide-character string to a window |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library ...] #include <curses.h> int addnwstr(const wchar_t *wstr, int n); int addwstr(const wchar_t *wstr); int mvaddnwstr(int y, int x, const wchar_t *wstr, int n); int mvaddwstr(int y, int x, const wchar_t *wstr); int mvwaddnwstr(WINDOW*win, int y, int x, const wchar_t *wstr, int n); int mvwaddwstr(WINDOW*win, int y, int x, const wchar_t *wstr); int waddnwstr(WINDOW*win, const wchar_t *wstr, int n); int waddwstr(WINDOW*win, const wchar_t *wstr);</pre> |
| DESCRIPTION | <p>The <code>addwstr()</code> function writes a null-terminated wide-character string to the <code>stdscr</code> window at the current cursor position. The <code>waddwstr()</code> function performs an identical action, but writes the string to the window specified by <code>win</code>. The <code>mvaddwstr()</code> and <code>mvwaddwstr()</code> functions write the string to the position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <code>win</code>).</p> <p>The <code>addnwstr()</code>, <code>waddnwstr()</code>, <code>mvaddnwstr()</code>, and <code>mvwaddnwstr()</code> functions write at most <code>n</code> characters to the window. If <code>n</code> is less than 0, the entire string is written. The former two functions place the characters at the current cursor position; the latter two commands use the position specified by the <code>x</code> and <code>y</code> parameters.</p> <p>All of these functions advance the cursor after writing the string.</p> <p>These functions are functionally equivalent to building a <code>cchar_t</code> from the <code>wchar_t</code> and the window rendition (or background character and rendition) and calling the <code>wadd_wch(3XCURSES)</code> function once for each <code>wchar_t</code> in the string. Refer to the <code>curses(3XCURSES)</code> man page for a complete description of special character handling and of the interaction between the window rendition (or background character and rendition) and the character written.</p> <p>Note that these functions differ from the <code>add_wchnstr(3XCURSES)</code> set of functions in that the latter copy the string as is (without combining each character with the foreground and background attributes of the window).</p> |
| PARAMETERS | <p><code>wstr</code> Is a pointer to the wide-character string that is to be written to the window.</p> |

waddwstr(3XCURSES)

n Is the maximum number of characters to be copied from *wstr*. If *n* is less than 0, the entire string is written or as much of it as fits on the line.

y Is the y (row) coordinate of the starting position of *wstr* in the window.

x Is the x (column) coordinate of the starting position of *wstr* in the window.

win Is a pointer to the window in which the string is to be written.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `add_wch(3XCURSES)`, `add_wchnstr(3XCURSES)`, `curses(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

wadjcurpos(3CURSES)

| NAME | <code> curs_alecompat, movenextch, wmovenextch, moveprevch, wmoveprevch, adjcurpos, wadjcurpos</code> – these functions are added to ALE curses library for moving the cursor by character. | | | | |
|---------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int movenextch(void); int wmovenextch(WINDOW *win); int moveprevch(void); int wmoveprevch(WINDOW *win); int adjcurpos(void); int wadjcurpos(WINDOW *win);</pre> | | | | |
| DESCRIPTION | <p><code>movenextch()</code> and <code>wmovenextch()</code> move the cursor to the next character to the right. If the next character is a multicolumn character, the cursor is positioned on the first (left-most) column of that character. The new cursor position will be on the next character, even if the cursor was originally positioned on the left-most column of a multicolumn character. Note that the simple cursor increment (<code>++x</code>) does not guarantee movement to the next character, if the cursor was originally positioned on a multicolumn character. <code>getyx(3CURSES)</code> can be used to find the new position.</p> <p><code>moveprevch()</code> and <code>wmoveprevch()</code> routines are the opposite of <code>movenextch()</code> and <code>wmovenextch()</code>, moving the cursor to the left-most column of the previous character.</p> <p><code>adjcurpos()</code> and <code>wadjcurpos()</code> move the cursor to the first(left-most) column of the multicolumn character that the cursor is presently on. If the cursor is already on the first column, or if the cursor is on a single-column character, these routines will have no effect.</p> | | | | |
| RETURN VALUE | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>getyx(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header file <code><curses.h></code> automatically includes the header files <code><stdio.h></code> , <code><unctrl.h></code> and <code><widec.h></code> . Note that <code>movenextch()</code> , <code>moveprevch()</code> , and <code>adjcurpos()</code> may be macros. | | | | |

| | | | | | | | | | |
|--------------------|--|--------------|--|--------------|---------------------------------------|-------------|-----------------------------|------------|---|
| NAME | attr_get, attr_off, attr_on, attr_set, color_set, wattr_get, wattr_off, wattr_on, wattr_set, wcolor_set – control window attributes | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int attr_get(attr_t *attrs, short *color, void *opts); int attr_off(attr_t attrs, void *opts); int attr_on(attr_t attrs, void *opts); int attr_set(attr_t attrs, short color, void *opts); int color_set(short *color, void *opts); int wattr_get(WINDOW *win, attr_t attrs, short *color, void *opts); int wattr_off(WINDOW *win, attr_t attrs, void *opts); int wattr_on(WINDOW *win, attr_t attrs, void *opts); int wattr_set(WINDOW *win, attr_t attrs, short color, void *opts); int wcolor_set(WINDOW *win, short color, void *opts);</pre> | | | | | | | | |
| DESCRIPTION | <p>The attr_get() function retrieves the current rendition of <i>stdscr</i>. The wattr_get() function retrieves the current rendition of window <i>win</i>. If <i>attrs</i> or <i>color</i> is a null pointer, no information is retrieved.</p> <p>The attr_off() and attr_on() functions unset and set, respectively, the specified window attributes of <i>stdscr</i>. These functions only affect the attributes specified; attributes that existed before the call are retained.</p> <p>The wattr_off() and wattr_on() functions unset or set the specified attributes for window <i>win</i>.</p> <p>The attr_set() and wattr_set() functions change the rendition of <i>stdscr</i> and <i>win</i>; the old values are not retained.</p> <p>The color_set() and wcolor_set() functions set the window color of <i>stdscr</i> and <i>win</i> to <i>color</i>.</p> <p>The attributes and color pairs that can be used are specified in the Attributes, Color Pairs, and Renditions section of the curses(3XCURSES) man page.</p> | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 10px;"><i>attrs</i></td> <td>Is a pointer to the foreground window attributes to be set or unset.</td> </tr> <tr> <td><i>color</i></td> <td>Is a pointer to a color pair number .</td> </tr> <tr> <td><i>opts</i></td> <td>Is reserved for future use.</td> </tr> <tr> <td><i>win</i></td> <td>Is a pointer to the window in which attribute changes are to be made.</td> </tr> </table> | <i>attrs</i> | Is a pointer to the foreground window attributes to be set or unset. | <i>color</i> | Is a pointer to a color pair number . | <i>opts</i> | Is reserved for future use. | <i>win</i> | Is a pointer to the window in which attribute changes are to be made. |
| <i>attrs</i> | Is a pointer to the foreground window attributes to be set or unset. | | | | | | | | |
| <i>color</i> | Is a pointer to a color pair number . | | | | | | | | |
| <i>opts</i> | Is reserved for future use. | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which attribute changes are to be made. | | | | | | | | |

wattr_get(3XCURSES)

RETURN VALUES These functions always return OK.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `add_wch(3XCURSES)`, `addnwstr(3XCURSES)`, `attroff(3XCURSES)`, `bkgrndset(3XCURSES)`, `curses(3XCURSES)`, `init_color(3XCURSES)`, `libcurses(3XCURSES)`, `start_color(3XCURSES)`, `attributes(5)`, `standards(5)`

| | | | | | | | | | | | | | | | |
|--------------------|--|------------|--|-------------|-------------|-----------|---------------|---------|----------|-------|-------------|--------|----------------------|--------------|-------------------------|
| NAME | curs_attr, attroff, wattroff, attron, wattron, attrset, wattrset, standend, wstandend, standout, wstandout – curses character and window attribute control routines | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int attroff(int attrs); int wattroff(WINDOW *win, int attrs); int attron(int attrs); int wattron(WINDOW *win, int attrs); int attrset(int attrs); int wattrset(WINDOW *win, int attrs); int standend(void); int wstandend(WINDOW *win); int standout(void); int wstandout(WINDOW *win);</pre> | | | | | | | | | | | | | | |
| DESCRIPTION | <p>All of these routines manipulate the current attributes of the named window. The current attributes of a window are applied to all characters that are written into the window with <code>waddch()</code>, <code>waddstr()</code>, and <code>wprintw()</code>. Attributes are a property of the character, and move with the character through any scrolling and insert/delete line/character operations. To the extent possible on the particular terminal, they are displayed as the graphic rendition of characters put on the screen.</p> <p>The routine <code>attrset()</code> sets the current attributes of the given window to <code>attrs</code>. The routine <code>attroff()</code> turns off the named attributes without turning any other attributes on or off. The routine <code>attron()</code> turns on the named attributes without affecting any others. The routine <code>standout()</code> is the same as <code>attron(A_STANDOUT)</code>. The routine <code>standend()</code> is the same as <code>attrset()</code>, that is, it turns off all attributes.</p> | | | | | | | | | | | | | | |
| Attributes | <p>The following video attributes, defined in <code><curses.h></code>, can be passed to the routines <code>attron()</code>, <code>attroff()</code>, and <code>attrset()</code>, or OR-ed with the characters passed to <code>addch()</code>.</p> <table border="0"> <tr> <td style="padding-right: 20px;">A_STANDOUT</td> <td>Best highlighting mode of the terminal</td> </tr> <tr> <td>A_UNDERLINE</td> <td>Underlining</td> </tr> <tr> <td>A_REVERSE</td> <td>Reverse video</td> </tr> <tr> <td>A_BLINK</td> <td>Blinking</td> </tr> <tr> <td>A_DIM</td> <td>Half bright</td> </tr> <tr> <td>A_BOLD</td> <td>Extra bright or bold</td> </tr> <tr> <td>A_ALTCHARSET</td> <td>Alternate character set</td> </tr> </table> | A_STANDOUT | Best highlighting mode of the terminal | A_UNDERLINE | Underlining | A_REVERSE | Reverse video | A_BLINK | Blinking | A_DIM | Half bright | A_BOLD | Extra bright or bold | A_ALTCHARSET | Alternate character set |
| A_STANDOUT | Best highlighting mode of the terminal | | | | | | | | | | | | | | |
| A_UNDERLINE | Underlining | | | | | | | | | | | | | | |
| A_REVERSE | Reverse video | | | | | | | | | | | | | | |
| A_BLINK | Blinking | | | | | | | | | | | | | | |
| A_DIM | Half bright | | | | | | | | | | | | | | |
| A_BOLD | Extra bright or bold | | | | | | | | | | | | | | |
| A_ALTCHARSET | Alternate character set | | | | | | | | | | | | | | |

wattroff(3CURSES)

A_CHARTEXT Bit-mask to extract a character

COLOR_PAIR(*n*) Color-pair number *n*

The following macro is the reverse of COLOR_PAIR(*n*) :

PAIR_NUMBER(*attrs*) Returns the pair number associated with the
COLOR_PAIR(*n*) attribute

RETURN VALUES These routines always return 1.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_addch(3CURSES), curs_addstr(3CURSES), curs_printw(3CURSES),
curses(3CURSES), attributes(5)

NOTES The header < curses . h > automatically includes the headers < stdio . h > and
< unctrl . h >.

Note that attroff () , wattroff () , attron () , wattron () , wattrset () ,
standend () , and standout () may be macros.

| | |
|--------------------|--|
| NAME | attr_get, attr_off, attr_on, attr_set, color_set, wattr_get, wattr_off, wattr_on, wattr_set, wcolor_set – control window attributes |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int attr_get(attr_t *attrs, short *color, void *opts); int attr_off(attr_t attrs, void *opts); int attr_on(attr_t attrs, void *opts); int attr_set(attr_t attrs, short color, void *opts); int color_set(short *color, void *opts); int wattr_get(WINDOW *win, attr_t attrs, short *color, void *opts); int wattr_off(WINDOW *win, attr_t attrs, void *opts); int wattr_on(WINDOW *win, attr_t attrs, void *opts); int wattr_set(WINDOW *win, attr_t attrs, short color, void *opts); int wcolor_set(WINDOW *win, short color, void *opts);</pre> |
| DESCRIPTION | <p>The attr_get() function retrieves the current rendition of <i>stdscr</i>. The wattr_get() function retrieves the current rendition of window <i>win</i>. If <i>attrs</i> or <i>color</i> is a null pointer, no information is retrieved.</p> <p>The attr_off() and attr_on() functions unset and set, respectively, the specified window attributes of <i>stdscr</i>. These functions only affect the attributes specified; attributes that existed before the call are retained.</p> <p>The wattr_off() and wattr_on() functions unset or set the specified attributes for window <i>win</i>.</p> <p>The attr_set() and wattr_set() functions change the rendition of <i>stdscr</i> and <i>win</i>; the old values are not retained.</p> <p>The color_set() and wcolor_set() functions set the window color of <i>stdscr</i> and <i>win</i> to <i>color</i>.</p> <p>The attributes and color pairs that can be used are specified in the Attributes, Color Pairs, and Renditions section of the curses(3XCURSES) man page.</p> |
| PARAMETERS | <p><i>attrs</i> Is a pointer to the foreground window attributes to be set or unset.</p> <p><i>color</i> Is a pointer to a color pair number .</p> <p><i>opts</i> Is reserved for future use.</p> <p><i>win</i> Is a pointer to the window in which attribute changes are to be made.</p> |

wattr_off(3XCURSES)

RETURN VALUES These functions always return OK.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `add_wch(3XCURSES)`, `addnwstr(3XCURSES)`, `attroff(3XCURSES)`, `bkgrndset(3XCURSES)`, `curses(3XCURSES)`, `init_color(3XCURSES)`, `libcurses(3XCURSES)`, `start_color(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|----------------------|--|
| NAME | attroff, attron, attrset, wattroff, wattron, wattrset – change foreground window attributes |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int attroff(int attrs); int attron(int attrs); int attrset(int attrs); int wattroff(WINDOW *win, int attrs); int wattron(WINDOW *win, int attrs); int wattrset(WINDOW *win, int attrs);</pre> |
| DESCRIPTION | <p>The attroff() and attron() functions unset and set, respectively, the specified window attributes of stdscr. These functions only affect the attributes specified; attributes that existed before the call are retained. The wattroff() and wattron() functions unset or set the specified attributes for window win.</p> <p>The attrset() and wattrset() functions change the specified window renditions of stdscr and win to new values; the old values are not retained.</p> <p>The attributes that can be used are specified in the Attributes, Color Pairs, and Renditions section of the curses(3XCURSES) man page.</p> <p>Here is an example that prints some text using the current window rendition, adds underlining, changes the attributes, prints more text, then changes the attributes back.</p> <pre>printw("This word is"); attron(A_UNDERLINE); printw("underlined."); attroff(A_NORMAL); printw("This is back to normal text.\n"); refresh();</pre> |
| PARAMETERS | <p><i>attrs</i> are the foreground window attributes to be set or unset.</p> <p><i>win</i> Is a pointer to the window in which attribute changes are to be made.</p> |
| RETURN VALUES | These functions always return OK or 1. |
| ERRORS | None. |
| USAGE | All of these functions may be macros. |

wattroff(3XCURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addch(3XCURSES)`, `addnstr(3XCURSES)`, `attr_get(3XCURSES)`, `bkgdset(3XCURSES)`, `curses(3XCURSES)`, `init_color(3XCURSES)`, `libcurses(3XCURSES)`, `start_color(3XCURSES)`, `attributes(5)`, `standards(5)`

| | | | | | | | | | | | | | | | |
|--------------------|--|------------|--|-------------|-------------|-----------|---------------|---------|----------|-------|-------------|--------|----------------------|--------------|-------------------------|
| NAME | curs_attr, attroff, wattroff, attron, wattron, attrset, wattrset, standend, wstandend, standout, wstandout – curses character and window attribute control routines | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int attroff(int attrs); int wattroff(WINDOW *win, int attrs); int attron(int attrs); int wattron(WINDOW *win, int attrs); int attrset(int attrs); int wattrset(WINDOW *win, int attrs); int standend(void); int wstandend(WINDOW *win); int standout(void); int wstandout(WINDOW *win);</pre> | | | | | | | | | | | | | | |
| DESCRIPTION | <p>All of these routines manipulate the current attributes of the named window. The current attributes of a window are applied to all characters that are written into the window with <code>waddch()</code>, <code>waddstr()</code>, and <code>wprintw()</code>. Attributes are a property of the character, and move with the character through any scrolling and insert/delete line/character operations. To the extent possible on the particular terminal, they are displayed as the graphic rendition of characters put on the screen.</p> <p>The routine <code>attrset()</code> sets the current attributes of the given window to <code>attrs</code>. The routine <code>attroff()</code> turns off the named attributes without turning any other attributes on or off. The routine <code>attron()</code> turns on the named attributes without affecting any others. The routine <code>standout()</code> is the same as <code>attron(A_STANDOUT)</code>. The routine <code>standend()</code> is the same as <code>attrset()</code>, that is, it turns off all attributes.</p> | | | | | | | | | | | | | | |
| Attributes | <p>The following video attributes, defined in <code><curses.h></code>, can be passed to the routines <code>attron()</code>, <code>attroff()</code>, and <code>attrset()</code>, or OR-ed with the characters passed to <code>addch()</code>.</p> <table border="0"> <tr> <td style="padding-right: 20px;">A_STANDOUT</td> <td>Best highlighting mode of the terminal</td> </tr> <tr> <td>A_UNDERLINE</td> <td>Underlining</td> </tr> <tr> <td>A_REVERSE</td> <td>Reverse video</td> </tr> <tr> <td>A_BLINK</td> <td>Blinking</td> </tr> <tr> <td>A_DIM</td> <td>Half bright</td> </tr> <tr> <td>A_BOLD</td> <td>Extra bright or bold</td> </tr> <tr> <td>A_ALTCHARSET</td> <td>Alternate character set</td> </tr> </table> | A_STANDOUT | Best highlighting mode of the terminal | A_UNDERLINE | Underlining | A_REVERSE | Reverse video | A_BLINK | Blinking | A_DIM | Half bright | A_BOLD | Extra bright or bold | A_ALTCHARSET | Alternate character set |
| A_STANDOUT | Best highlighting mode of the terminal | | | | | | | | | | | | | | |
| A_UNDERLINE | Underlining | | | | | | | | | | | | | | |
| A_REVERSE | Reverse video | | | | | | | | | | | | | | |
| A_BLINK | Blinking | | | | | | | | | | | | | | |
| A_DIM | Half bright | | | | | | | | | | | | | | |
| A_BOLD | Extra bright or bold | | | | | | | | | | | | | | |
| A_ALTCHARSET | Alternate character set | | | | | | | | | | | | | | |

wattron(3CURSES)

A_CHARTEXT Bit-mask to extract a character

COLOR_PAIR(*n*) Color-pair number *n*

The following macro is the reverse of COLOR_PAIR(*n*) :

PAIR_NUMBER(*attrs*) Returns the pair number associated with the
COLOR_PAIR(*n*) attribute

RETURN VALUES These routines always return 1.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_addch(3CURSES), curs_addstr(3CURSES), curs_printw(3CURSES),
curses(3CURSES), attributes(5)

NOTES The header < curses.h > automatically includes the headers < stdio.h > and
< unctrl.h >.

Note that attroff(), wattroff(), attron(), wattron(), wattrset(),
standend(), and standout() may be macros.

| | | | | | | | | | |
|--------------------|--|--------------|--|--------------|---------------------------------------|-------------|-----------------------------|------------|---|
| NAME | attr_get, attr_off, attr_on, attr_set, color_set, wattr_get, wattr_off, wattr_on, wattr_set, wcolor_set – control window attributes | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int attr_get(attr_t *attrs, short *color, void *opts); int attr_off(attr_t attrs, void *opts); int attr_on(attr_t attrs, void *opts); int attr_set(attr_t attrs, short color, void *opts); int color_set(short *color, void *opts); int wattr_get(WINDOW *win, attr_t attrs, short *color, void *opts); int wattr_off(WINDOW *win, attr_t attrs, void *opts); int wattr_on(WINDOW *win, attr_t attrs, void *opts); int wattr_set(WINDOW *win, attr_t attrs, short color, void *opts); int wcolor_set(WINDOW *win, short color, void *opts);</pre> | | | | | | | | |
| DESCRIPTION | <p>The attr_get() function retrieves the current rendition of <i>stdscr</i>. The wattr_get() function retrieves the current rendition of window <i>win</i>. If <i>attrs</i> or <i>color</i> is a null pointer, no information is retrieved.</p> <p>The attr_off() and attr_on() functions unset and set, respectively, the specified window attributes of <i>stdscr</i>. These functions only affect the attributes specified; attributes that existed before the call are retained.</p> <p>The wattr_off() and wattr_on() functions unset or set the specified attributes for window <i>win</i>.</p> <p>The attr_set() and wattr_set() functions change the rendition of <i>stdscr</i> and <i>win</i>; the old values are not retained.</p> <p>The color_set() and wcolor_set() functions set the window color of <i>stdscr</i> and <i>win</i> to <i>color</i>.</p> <p>The attributes and color pairs that can be used are specified in the Attributes, Color Pairs, and Renditions section of the curses(3XCURSES) man page.</p> | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 10px;"><i>attrs</i></td> <td>Is a pointer to the foreground window attributes to be set or unset.</td> </tr> <tr> <td><i>color</i></td> <td>Is a pointer to a color pair number .</td> </tr> <tr> <td><i>opts</i></td> <td>Is reserved for future use.</td> </tr> <tr> <td><i>win</i></td> <td>Is a pointer to the window in which attribute changes are to be made.</td> </tr> </table> | <i>attrs</i> | Is a pointer to the foreground window attributes to be set or unset. | <i>color</i> | Is a pointer to a color pair number . | <i>opts</i> | Is reserved for future use. | <i>win</i> | Is a pointer to the window in which attribute changes are to be made. |
| <i>attrs</i> | Is a pointer to the foreground window attributes to be set or unset. | | | | | | | | |
| <i>color</i> | Is a pointer to a color pair number . | | | | | | | | |
| <i>opts</i> | Is reserved for future use. | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which attribute changes are to be made. | | | | | | | | |

wattr_on(3XCURSES)

RETURN VALUES These functions always return OK.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `add_wch(3XCURSES)`, `addnwstr(3XCURSES)`, `attroff(3XCURSES)`, `bkgrndset(3XCURSES)`, `curses(3XCURSES)`, `init_color(3XCURSES)`, `libcurses(3XCURSES)`, `start_color(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|----------------------|--|
| NAME | attroff, attron, attrset, wattroff, wattron, wattrset – change foreground window attributes |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int attroff(int attrs); int attron(int attrs); int attrset(int attrs); int wattroff(WINDOW *win, int attrs); int wattron(WINDOW *win, int attrs); int wattrset(WINDOW *win, int attrs);</pre> |
| DESCRIPTION | <p>The attroff() and attron() functions unset and set, respectively, the specified window attributes of stdscr. These functions only affect the attributes specified; attributes that existed before the call are retained. The wattroff() and wattron() functions unset or set the specified attributes for window win.</p> <p>The attrset() and wattrset() functions change the specified window renditions of stdscr and win to new values; the old values are not retained.</p> <p>The attributes that can be used are specified in the Attributes, Color Pairs, and Renditions section of the curses(3XCURSES) man page.</p> <p>Here is an example that prints some text using the current window rendition, adds underlining, changes the attributes, prints more text, then changes the attributes back.</p> <pre>printw("This word is"); attron(A_UNDERLINE); printw("underlined."); attroff(A_NORMAL); printw("This is back to normal text.\n"); refresh();</pre> |
| PARAMETERS | <p><i>attrs</i> are the foreground window attributes to be set or unset.</p> <p><i>win</i> Is a pointer to the window in which attribute changes are to be made.</p> |
| RETURN VALUES | These functions always return OK or 1. |
| ERRORS | None. |
| USAGE | All of these functions may be macros. |

wattron(3XCURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addch(3XCURSES)`, `addnstr(3XCURSES)`, `attr_get(3XCURSES)`, `bkgdset(3XCURSES)`, `curses(3XCURSES)`, `init_color(3XCURSES)`, `libcurses(3XCURSES)`, `start_color(3XCURSES)`, `attributes(5)`, `standards(5)`

| | | | | | | | | | | | | | | | |
|--------------------|--|------------|--|-------------|-------------|-----------|---------------|---------|----------|-------|-------------|--------|----------------------|--------------|-------------------------|
| NAME | curs_attr, attroff, wattroff, attron, wattron, attrset, wattrset, standend, wstandend, standout, wstandout – curses character and window attribute control routines | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int attroff(int attrs); int wattroff(WINDOW *win, int attrs); int attron(int attrs); int wattron(WINDOW *win, int attrs); int attrset(int attrs); int wattrset(WINDOW *win, int attrs); int standend(void); int wstandend(WINDOW *win); int standout(void); int wstandout(WINDOW *win);</pre> | | | | | | | | | | | | | | |
| DESCRIPTION | <p>All of these routines manipulate the current attributes of the named window. The current attributes of a window are applied to all characters that are written into the window with <code>waddch()</code>, <code>waddstr()</code>, and <code>wprintw()</code>. Attributes are a property of the character, and move with the character through any scrolling and insert/delete line/character operations. To the extent possible on the particular terminal, they are displayed as the graphic rendition of characters put on the screen.</p> <p>The routine <code>attrset()</code> sets the current attributes of the given window to <code>attrs</code>. The routine <code>attroff()</code> turns off the named attributes without turning any other attributes on or off. The routine <code>attron()</code> turns on the named attributes without affecting any others. The routine <code>standout()</code> is the same as <code>attron(A_STANDOUT)</code>. The routine <code>standend()</code> is the same as <code>attrset()</code>, that is, it turns off all attributes.</p> | | | | | | | | | | | | | | |
| Attributes | <p>The following video attributes, defined in <code><curses.h></code>, can be passed to the routines <code>attron()</code>, <code>attroff()</code>, and <code>attrset()</code>, or OR-ed with the characters passed to <code>addch()</code>.</p> <table border="0"> <tr> <td style="padding-right: 20px;">A_STANDOUT</td> <td>Best highlighting mode of the terminal</td> </tr> <tr> <td>A_UNDERLINE</td> <td>Underlining</td> </tr> <tr> <td>A_REVERSE</td> <td>Reverse video</td> </tr> <tr> <td>A_BLINK</td> <td>Blinking</td> </tr> <tr> <td>A_DIM</td> <td>Half bright</td> </tr> <tr> <td>A_BOLD</td> <td>Extra bright or bold</td> </tr> <tr> <td>A_ALTCHARSET</td> <td>Alternate character set</td> </tr> </table> | A_STANDOUT | Best highlighting mode of the terminal | A_UNDERLINE | Underlining | A_REVERSE | Reverse video | A_BLINK | Blinking | A_DIM | Half bright | A_BOLD | Extra bright or bold | A_ALTCHARSET | Alternate character set |
| A_STANDOUT | Best highlighting mode of the terminal | | | | | | | | | | | | | | |
| A_UNDERLINE | Underlining | | | | | | | | | | | | | | |
| A_REVERSE | Reverse video | | | | | | | | | | | | | | |
| A_BLINK | Blinking | | | | | | | | | | | | | | |
| A_DIM | Half bright | | | | | | | | | | | | | | |
| A_BOLD | Extra bright or bold | | | | | | | | | | | | | | |
| A_ALTCHARSET | Alternate character set | | | | | | | | | | | | | | |

wattrset(3CURSES)

A_CHARTEXT Bit-mask to extract a character

COLOR_PAIR(*n*) Color-pair number *n*

The following macro is the reverse of COLOR_PAIR(*n*) :

PAIR_NUMBER(*attrs*) Returns the pair number associated with the
COLOR_PAIR(*n*) attribute

RETURN VALUES These routines always return 1.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_addch(3CURSES), curs_addstr(3CURSES), curs_printw(3CURSES),
curses(3CURSES), attributes(5)

NOTES The header <curses.h> automatically includes the headers <stdio.h> and
<unctrl.h>.

Note that attroff(), wattroff(), attron(), wattron(), wattrset(),
standend(), and standout() may be macros.

| | | | | | | | | | |
|--------------------|--|--------------|--|--------------|---------------------------------------|-------------|-----------------------------|------------|---|
| NAME | attr_get, attr_off, attr_on, attr_set, color_set, wattr_get, wattr_off, wattr_on, wattr_set, wcolor_set – control window attributes | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int attr_get(attr_t *attrs, short *color, void *opts); int attr_off(attr_t attrs, void *opts); int attr_on(attr_t attrs, void *opts); int attr_set(attr_t attrs, short color, void *opts); int color_set(short *color, void *opts); int wattr_get(WINDOW *win, attr_t attrs, short *color, void *opts); int wattr_off(WINDOW *win, attr_t attrs, void *opts); int wattr_on(WINDOW *win, attr_t attrs, void *opts); int wattr_set(WINDOW *win, attr_t attrs, short color, void *opts); int wcolor_set(WINDOW *win, short color, void *opts);</pre> | | | | | | | | |
| DESCRIPTION | <p>The attr_get() function retrieves the current rendition of <i>stdscr</i>. The wattr_get() function retrieves the current rendition of window <i>win</i>. If <i>attrs</i> or <i>color</i> is a null pointer, no information is retrieved.</p> <p>The attr_off() and attr_on() functions unset and set, respectively, the specified window attributes of <i>stdscr</i>. These functions only affect the attributes specified; attributes that existed before the call are retained.</p> <p>The wattr_off() and wattr_on() functions unset or set the specified attributes for window <i>win</i>.</p> <p>The attr_set() and wattr_set() functions change the rendition of <i>stdscr</i> and <i>win</i>; the old values are not retained.</p> <p>The color_set() and wcolor_set() functions set the window color of <i>stdscr</i> and <i>win</i> to <i>color</i>.</p> <p>The attributes and color pairs that can be used are specified in the Attributes, Color Pairs, and Renditions section of the curses(3XCURSES) man page.</p> | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 10px;"><i>attrs</i></td> <td>Is a pointer to the foreground window attributes to be set or unset.</td> </tr> <tr> <td style="padding-right: 10px;"><i>color</i></td> <td>Is a pointer to a color pair number .</td> </tr> <tr> <td style="padding-right: 10px;"><i>opts</i></td> <td>Is reserved for future use.</td> </tr> <tr> <td style="padding-right: 10px;"><i>win</i></td> <td>Is a pointer to the window in which attribute changes are to be made.</td> </tr> </table> | <i>attrs</i> | Is a pointer to the foreground window attributes to be set or unset. | <i>color</i> | Is a pointer to a color pair number . | <i>opts</i> | Is reserved for future use. | <i>win</i> | Is a pointer to the window in which attribute changes are to be made. |
| <i>attrs</i> | Is a pointer to the foreground window attributes to be set or unset. | | | | | | | | |
| <i>color</i> | Is a pointer to a color pair number . | | | | | | | | |
| <i>opts</i> | Is reserved for future use. | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which attribute changes are to be made. | | | | | | | | |

wattr_set(3XCURSES)

RETURN VALUES These functions always return OK.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `add_wch(3XCURSES)`, `addnwstr(3XCURSES)`, `attroff(3XCURSES)`, `bkgrndset(3XCURSES)`, `curses(3XCURSES)`, `init_color(3XCURSES)`, `libcurses(3XCURSES)`, `start_color(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|----------------------|--|
| NAME | attroff, attron, attrset, wattroff, wattron, wattrset – change foreground window attributes |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int attroff(int attrs); int attron(int attrs); int attrset(int attrs); int wattroff(WINDOW *win, int attrs); int wattron(WINDOW *win, int attrs); int wattrset(WINDOW *win, int attrs);</pre> |
| DESCRIPTION | <p>The <code>attroff()</code> and <code>attron()</code> functions unset and set, respectively, the specified window attributes of <code>stdscr</code>. These functions only affect the attributes specified; attributes that existed before the call are retained. The <code>wattroff()</code> and <code>wattron()</code> functions unset or set the specified attributes for window <i>win</i>.</p> <p>The <code>attrset()</code> and <code>wattrset()</code> functions change the specified window renditions of <code>stdscr</code> and <i>win</i> to new values; the old values are not retained.</p> <p>The attributes that can be used are specified in the <i>Attributes, Color Pairs, and Renditions</i> section of the <code>curses(3XCURSES)</code> man page.</p> <p>Here is an example that prints some text using the current window rendition, adds underlining, changes the attributes, prints more text, then changes the attributes back.</p> <pre>printw("This word is"); attron(A_UNDERLINE); printw("underlined."); attroff(A_NORMAL); printw("This is back to normal text.\n"); refresh();</pre> |
| PARAMETERS | <p><i>attrs</i> are the foreground window attributes to be set or unset.</p> <p><i>win</i> Is a pointer to the window in which attribute changes are to be made.</p> |
| RETURN VALUES | These functions always return OK or 1. |
| ERRORS | None. |
| USAGE | All of these functions may be macros. |

wattrset(3XCURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `addch(3XCURSES)`, `addnstr(3XCURSES)`, `attr_get(3XCURSES)`, `bkgdset(3XCURSES)`, `curses(3XCURSES)`, `init_color(3XCURSES)`, `libcurses(3XCURSES)`, `start_color(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_bkgd, bkgd, bkgdset, wbkgdset, wbkgd – curses window background manipulation routines | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int bkgd(chtype ch); void bkgdset(chtype ch); void wbkgdset(WINDOW *win, chtype ch); int wbkgd(WINDOW *win, chtype ch);</pre> | | | | |
| DESCRIPTION | <p>The <code>bkgdsets()</code> and <code>wbkgdset()</code> routines manipulate the background of the named window. Background is a <code>chtype</code> consisting of any combination of attributes and a character. The attribute part of the background is combined (ORed) with all non-blank characters that are written into the window with <code>waddch()</code>. Both the character and attribute parts of the background are combined with the blank characters. The background becomes a property of the character and moves with the character through any scrolling and insert/delete line/character operations. To the extent possible on a particular terminal, the attribute part of the background is displayed as the graphic rendition of the character put on the screen.</p> <p>The <code>bkgd()</code> and <code>wbkgd()</code> routines combine the new background with every position in the window. Background is any combination of attributes and a character. Only the attribute part is used to set the background of non-blank characters, while both character and attributes are used for blank positions. To the extent possible on a particular terminal, the attribute part of the background is displayed as the graphic rendition of the character put on the screen.</p> | | | | |
| RETURN VALUES | <code>bkgd()</code> and <code>wbkgd()</code> return the integer <code>OK</code> , or a non-negative integer, if <code>immedok()</code> is set. See <code>curs_ouptops(3CURSES)</code> . | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_addch(3CURSES)</code> , <code>curs_ouptops(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that <code>bkgdset()</code> and <code>bkgd()</code> may be macros.</p> | | | | |

wbkgd(3XCURSES)

| | |
|----------------------|---|
| NAME | <code>bkgd</code> , <code>bkgdset</code> , <code>getbkgd</code> , <code>wbkgd</code> , <code>wbkgdset</code> – set or get the background character (and rendition) of window |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int bkgd(chtype <i>ch</i>); void bkgdset(chtype <i>ch</i>); chtype getbkgd(WINDOW *<i>win</i>); int wbkgd(WINDOW *<i>win</i>, chtype <i>ch</i>); void wbkgdset(WINDOW *<i>win</i>, chtype <i>ch</i>);</pre> |
| DESCRIPTION | <p>The <code>bkgdset()</code> and <code>wbkgdset()</code> functions turn off the previous background attributes, logical OR the requested attributes into the window rendition, and set the background property of the current or specified window based on the information in <i>ch</i>. If <i>ch</i> refers to a multi-column character, the results are undefined.</p> <p>The <code>bkgd()</code> and <code>wbkgd()</code> functions turn off the previous background attributes, logical OR the requested attributes into the window rendition, and set the background property of the current or specified window and then apply this setting to every character position in that window:</p> <ul style="list-style-type: none">■ The rendition of every character on the screen is changed to the new window rendition.■ Wherever the former background character appears, it is changed to the new background character. <p>The <code>getbkgd()</code> function extracts the specified window's background character and rendition.</p> |
| PARAMETERS | <p><i>ch</i> Is the background character to be set.</p> <p><i>win</i> Is a pointer to the window in which the background character is to be set.</p> |
| RETURN VALUES | <p>Upon successful completion, the <code>bkgd()</code> and <code>wbkgd()</code> functions return OK. Otherwise, they return ERR.</p> <p>The <code>bkgdset()</code> and <code>wbkgdset()</code> functions do not return a value.</p> <p>Upon successful completion, the <code>getbkgd()</code> function returns the specified window's background character and rendition. Otherwise, it returns (chtype) ERR.</p> |
| ERRORS | No errors are defined. |

wbkgd(3XCURSES)

USAGE These functions are only guaranteed to operate reliably on character sets in which each character fits into a single byte, whose attributes can be expressed using only constants with the A_ prefix.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO addch(3XCURSES), addchstr(3XCURSES), attroff(3XCURSES), bkgrnd(3XCURSES), clear(3XCURSES), clrtoeol(3XCURSES), clrtoeol(3XCURSES), clrtobot(3XCURSES), erase(3XCURSES), inch(3XCURSES), libcurses(3XCURSES), mvprintw(3XCURSES), attributes(5), standards(5)

wbkgdset(3CURSES)

| NAME | <code> curs_bkgd, bkgd, bkgdset, wbkgdset, wbkgd</code> – curses window background manipulation routines | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int bkgd(chtype <i>ch</i>); void bkgdset(chtype <i>ch</i>); void wbkgdset(WINDOW *<i>win</i>, chtype <i>ch</i>); int wbkgd(WINDOW *<i>win</i>, chtype <i>ch</i>);</pre> | | | | |
| DESCRIPTION | <p>The <code>bkgdsets()</code> and <code>wbkgdset()</code> routines manipulate the background of the named window. Background is a <code>chtype</code> consisting of any combination of attributes and a character. The attribute part of the background is combined (ORed) with all non-blank characters that are written into the window with <code>waddch()</code>. Both the character and attribute parts of the background are combined with the blank characters. The background becomes a property of the character and moves with the character through any scrolling and insert/delete line/character operations. To the extent possible on a particular terminal, the attribute part of the background is displayed as the graphic rendition of the character put on the screen.</p> <p>The <code>bkgd()</code> and <code>wbkgd()</code> routines combine the new background with every position in the window. Background is any combination of attributes and a character. Only the attribute part is used to set the background of non-blank characters, while both character and attributes are used for blank positions. To the extent possible on a particular terminal, the attribute part of the background is displayed as the graphic rendition of the character put on the screen.</p> | | | | |
| RETURN VALUES | <code>bkgd()</code> and <code>wbkgd()</code> return the integer <code>OK</code> , or a non-negative integer, if <code>immedok()</code> is set. See <code>curls_ouptos(3CURSES)</code> . | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curls_addch(3CURSES)</code> , <code>curls_ouptos(3CURSES)</code> , <code>curls(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that <code>bkgdset()</code> and <code>bkgd()</code> may be macros.</p> | | | | |

| | |
|----------------------|--|
| NAME | bkgd, bkgdset, getbkgd, wbkgd, wbkgdset – set or get the background character (and rendition) of window |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int bkgd (chtype ch); void bkgdset (chtype ch); chtype getbkgd (WINDOW *win); int wbkgd (WINDOW *win, chtype ch); void wbkgdset (WINDOW *win, chtype ch);</pre> |
| DESCRIPTION | <p>The bkgdset () and wbkgdset () functions turn off the previous background attributes, logical OR the requested attributes into the window rendition, and set the background property of the current or specified window based on the information in <i>ch</i>. If <i>ch</i> refers to a multi-column character, the results are undefined.</p> <p>The bkgd () and wbkgd () functions turn off the previous background attributes, logical OR the requested attributes into the window rendition, and set the background property of the current or specified window and then apply this setting to every character position in that window:</p> <ul style="list-style-type: none"> ■ The rendition of every character on the screen is changed to the new window rendition. ■ Wherever the former background character appears, it is changed to the new background character. <p>The getbkgd () function extracts the specified window's background character and rendition.</p> |
| PARAMETERS | <p><i>ch</i> Is the background character to be set.</p> <p><i>win</i> Is a pointer to the window in which the background character is to be set.</p> |
| RETURN VALUES | <p>Upon successful completion, the bkgd () and wbkgd () functions return OK. Otherwise, they return ERR.</p> <p>The bkgdset () and wbkgdset () functions do not return a value.</p> <p>Upon successful completion, the getbkgd () function returns the specified window's background character and rendition. Otherwise, it returns (chtype) ERR.</p> |
| ERRORS | No errors are defined. |

wbkgdset(3XCURSES)

USAGE | These functions are only guaranteed to operate reliably on character sets in which each character fits into a single byte, whose attributes can be expressed using only constants with the A_ prefix.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | `addch(3XCURSES)`, `addchstr(3XCURSES)`, `attroff(3XCURSES)`, `bkgrnd(3XCURSES)`, `clear(3XCURSES)`, `clrtoeol(3XCURSES)`, `clrtobot(3XCURSES)`, `erase(3XCURSES)`, `inch(3XCURSES)`, `libcurses(3XCURSES)`, `mvprintw(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|----------------------|---|
| NAME | bkgrnd, bkgrndset, getbkgrnd, wbkgrnd, wbkgrndset, wgetbkgrnd – set or get the background character (and rendition) of window using a complex character |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int bkgrnd(const cchar_t *wch); void bkgrndset(const cchar_t *wch); int getbkgrnd(cchar_t *wch); int wbkgrnd(WINDOW *win, const cchar_t *wch); void wbkgrndset(WINDOW *win, const cchar_t *wch); int wgetbkgrnd(WINDOW *win, cchar_t *wch);</pre> |
| DESCRIPTION | <p>The bkgrndset () and wbkgrndset () functions turn off the previous background attributes, logical OR the requested attributes into the window rendition, and set the background property of the current or specified window based on the information in <i>wch</i>.</p> <p>The bkgrnd () and wbkgrnd () functions turn off the previous background attributes, logical OR the requested attributes into the window rendition, and set the background property of the current or specified window and then apply this setting to every character position in that window:</p> <ul style="list-style-type: none"> ■ The rendition of every character on the screen is changed to the new window rendition. ■ Wherever the former background character appears, it is changed to the new background character. <p>If <i>wch</i> refers to a non-spacing complex character for bkgrnd (), bkgrndset (), wbkgrnd (), and wbkgrndset (), then <i>wch</i> is added to the existing spacing complex character that is the background character. If <i>wch</i> refers to a multi-column character, the results are unspecified.</p> <p>The getbkgrnd () and wgetbkgrnd () functions store, into the area pointed to by <i>wch</i>, the window's background character and rendition.</p> |
| PARAMETERS | <p><i>wch</i> Is a pointer to the complex background character to be set.</p> <p><i>win</i> Is a pointer to the window in which the complex background character is to be set.</p> |
| RETURN VALUES | <p>The bkgrndset () and wbkgrndset () functions do not return a value.</p> <p>Upon successful completion, the other functions return OK. Otherwise, they return ERR.</p> |

wbkgrnd(3XCURSES)

ERRORS No errors are defined.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `add_wch(3XCURSES)`, `add_wchnstr(3XCURSES)`, `addch(3XCURSES)`, `addchstr(3XCURSES)`, `attroff(3XCURSES)`, `bkgd(3XCURSES)`, `clear(3XCURSES)`, `clrtoeol(3XCURSES)`, `clrtobot(3XCURSES)`, `erase(3XCURSES)`, `inch(3XCURSES)`, `libcurses(3XCURSES)`, `mvprintw(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|----------------------|---|
| NAME | bkgrnd, bkgrndset, getbkgrnd, wbkgrnd, wbkgrndset, wgetbkgrnd – set or get the background character (and rendition) of window using a complex character |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int bkgrnd(const cchar_t *wch); void bkgrndset(const cchar_t *wch); int getbkgrnd(cchar_t *wch); int wbkgrnd(WINDOW *win, const cchar_t *wch); void wbkgrndset(WINDOW *win, const cchar_t *wch); int wgetbkgrnd(WINDOW *win, cchar_t *wch);</pre> |
| DESCRIPTION | <p>The <code>bkgrndset()</code> and <code>wbkgrndset()</code> functions turn off the previous background attributes, logical OR the requested attributes into the window rendition, and set the background property of the current or specified window based on the information in <code>wch</code>.</p> <p>The <code>bkgrnd()</code> and <code>wbkgrnd()</code> functions turn off the previous background attributes, logical OR the requested attributes into the window rendition, and set the background property of the current or specified window and then apply this setting to every character position in that window:</p> <ul style="list-style-type: none"> ■ The rendition of every character on the screen is changed to the new window rendition. ■ Wherever the former background character appears, it is changed to the new background character. <p>If <code>wch</code> refers to a non-spacing complex character for <code>bkgrnd()</code>, <code>bkgrndset()</code>, <code>wbkgrnd()</code>, and <code>wbkgrndset()</code>, then <code>wch</code> is added to the existing spacing complex character that is the background character. If <code>wch</code> refers to a multi-column character, the results are unspecified.</p> <p>The <code>getbkgrnd()</code> and <code>wgetbkgrnd()</code> functions store, into the area pointed to by <code>wch</code>, the window's background character and rendition.</p> |
| PARAMETERS | <p><code>wch</code> Is a pointer to the complex background character to be set.</p> <p><code>win</code> Is a pointer to the window in which the complex background character is to be set.</p> |
| RETURN VALUES | <p>The <code>bkgrndset()</code> and <code>wbkgrndset()</code> functions do not return a value.</p> <p>Upon successful completion, the other functions return <code>OK</code>. Otherwise, they return <code>ERR</code>.</p> |

wbkgndset(3XCURSES)

ERRORS | No errors are defined.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | `add_wch(3XCURSES)`, `add_wchnstr(3XCURSES)`, `addch(3XCURSES)`, `addchstr(3XCURSES)`, `attroff(3XCURSES)`, `bkgd(3XCURSES)`, `clear(3XCURSES)`, `clrtoeol(3XCURSES)`, `clrtobot(3XCURSES)`, `erase(3XCURSES)`, `inch(3XCURSES)`, `libcurses(3XCURSES)`, `mvprintw(3XCURSES)`, `attributes(5)`, `standards(5)`

| | | | | | | | | | | | | | | | | | |
|--------------------|---|-----------|-------------------------|-----------|--------------------------|-----------|------------------------|-----------|---------------------------|-----------|----------------------|-----------|-----------------------|-----------|-------------------------|-----------|--------------------------|
| NAME | curs_border, border, wborder, box, whline, wvline – create curses borders, horizontal and vertical lines | | | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int border(chtype <i>ls</i>, chtype <i>rs</i>, chtype <i>ts</i>, chtype <i>bs</i>, chtype <i>tl</i>, chtype <i>tr</i>, chtype <i>bl</i>, chtype <i>br</i>); int wborder(WINDOW *<i>win</i>, chtype <i>ls</i>, chtype <i>rs</i>, chtype <i>ts</i>, chtype <i>bs</i>, chtype <i>tl</i>, chtype <i>tr</i>, chtype <i>bl</i>, chtype <i>br</i>); int box(WINDOW *<i>win</i>, chtype <i>verch</i>, chtype <i>horch</i>); int hline(chtype <i>ch</i>, int <i>n</i>); int whline(WINDOW *<i>win</i>, chtype <i>ch</i>, int <i>n</i>); int vline(chtype <i>ch</i>, int <i>n</i>); int wvline(WINDOW *<i>win</i>, chtype <i>ch</i>, int <i>n</i>);</pre> | | | | | | | | | | | | | | | | |
| DESCRIPTION | <p>With the <code>border()</code>, <code>wborder()</code>, and <code>box()</code> routines, a border is drawn around the edges of the window. The arguments and attributes are:</p> <hr/> <table border="0"> <tr> <td style="padding-right: 20px;"><i>ls</i></td> <td>left side of the border</td> </tr> <tr> <td><i>rs</i></td> <td>right side of the border</td> </tr> <tr> <td><i>ts</i></td> <td>top side of the border</td> </tr> <tr> <td><i>bs</i></td> <td>bottom side of the border</td> </tr> <tr> <td><i>tl</i></td> <td>top left-hand corner</td> </tr> <tr> <td><i>tr</i></td> <td>top right-hand corner</td> </tr> <tr> <td><i>bl</i></td> <td>bottom left-hand corner</td> </tr> <tr> <td><i>br</i></td> <td>bottom right-hand corner</td> </tr> </table> <hr/> <p>If any of these arguments is zero, then the following default values (defined in <code><curses.h></code>) are used respectively instead: <code>ACS_VLINE</code>, <code>ACS_VLINE</code>, <code>ACS_HLINE</code>, <code>ACS_HLINE</code>, <code>ACS_ULCORNER</code>, <code>ACS_URCORNER</code>, <code>ACS_BLCORNER</code>, <code>ACS_BRCORNER</code>.</p> <p><code>box(<i>win</i>, <i>verch</i>, <i>horch</i>)</code> is a shorthand for the following call:</p> <pre>wborder(<i>win</i>, <i>verch</i>, <i>verch</i>, <i>horch</i>, <i>horch</i>, 0, 0, 0, 0)</pre> <p><code>hline()</code> and <code>whline()</code> draw a horizontal (left to right) line using <i>ch</i> starting at the current cursor position in the window. The current cursor position is not changed. The line is at most <i>n</i> characters long, or as many as fit into the window.</p> | <i>ls</i> | left side of the border | <i>rs</i> | right side of the border | <i>ts</i> | top side of the border | <i>bs</i> | bottom side of the border | <i>tl</i> | top left-hand corner | <i>tr</i> | top right-hand corner | <i>bl</i> | bottom left-hand corner | <i>br</i> | bottom right-hand corner |
| <i>ls</i> | left side of the border | | | | | | | | | | | | | | | | |
| <i>rs</i> | right side of the border | | | | | | | | | | | | | | | | |
| <i>ts</i> | top side of the border | | | | | | | | | | | | | | | | |
| <i>bs</i> | bottom side of the border | | | | | | | | | | | | | | | | |
| <i>tl</i> | top left-hand corner | | | | | | | | | | | | | | | | |
| <i>tr</i> | top right-hand corner | | | | | | | | | | | | | | | | |
| <i>bl</i> | bottom left-hand corner | | | | | | | | | | | | | | | | |
| <i>br</i> | bottom right-hand corner | | | | | | | | | | | | | | | | |

wborder(3CURSES)

vline() and wvline() draw a vertical (top to bottom) line using *ch* starting at the current cursor position in the window. The current cursor position is not changed. The line is at most *n* characters long, or as many as fit into the window.

RETURN VALUES All routines return the integer OK, or a non-negative integer if `immedok()` is set. See `curls_outopts(3CURSES)`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curls_outopts(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `border()` and `box()` may be macros.

NAME border, box, wborder – add a single-byte border to a window

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

int border(chtype ls, chtype rs, chtype ts, chtype bs, chtype tl,
           chtype tr, chtype bl, chtype br);

int wborder(WINDOW *win, chtype ls, chtype rs, chtype ts, chtype bs,
            chtype tl, chtype tr, chtype bl, chtype br);

int box(WINDOW *win, chtype verch, chtype horch);
```

DESCRIPTION The `border()` and `wborder()` functions draw a border around the specified window. All parameters must be single-byte characters whose rendition can be expressed using only constants beginning with `ACS_`. A parameter with the value of 0 is replaced by the default value.

| Constant Values for Borders | | |
|-----------------------------|------------------|-------------------|
| Parameter | Default Constant | Default Character |
| <i>verch</i> | ACS_VLINE | |
| <i>horch</i> | ACS_HLINE | - |
| <i>ls</i> | ACS_VLINE | |
| <i>rs</i> | ACS_VLINE | |
| <i>ts</i> | ACS_HLINE | - |
| <i>bs</i> | ACS_HLINE | - |
| <i>bl</i> | ACS_BLCORNER | + |
| <i>br</i> | ACS_BRCORNER | + |
| <i>tl</i> | ACS_ULCORNER | + |
| <i>tr</i> | ACS_URCORNER | + |

The call

`box(win,
verch, horch)` is a short form for

```
wborder(win,
        verch, verch,
        horch, horch, 0, 0, 0,
```

wborder(3XCURSES)

o) When the window is boxed, the bottom and top rows and right and left columns overwrite existing text.

PARAMETERS

ls Is the character and rendition used for the left side of the border.
rs Is the character and rendition used for the right side of the border.
ts Is the character and rendition used for the top of the border.
bs Is the character and rendition used for the bottom of the border.
tl Is the character and rendition used for the top-left corner of the border.
tr Is the character and rendition used for the top-right corner of the border.
bl Is the character and rendition used for the bottom-left corner of the border.
br Is the character and rendition used for the bottom-right corner of the border.
win Is the pointer to the window in which the border or box is to be drawn.
verch Is the character and rendition used for the left and right columns of the box.
horch Is the character and rendition used for the top and bottom rows of the box.

RETURN VALUES

On success, these functions return OK. Otherwise, they return ERR.

ERRORS

None.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO

`add_wch(3XCURSES)`, `addch(3XCURSES)`, `attr_get(3XCURSES)`,
`attroff(3XCURSES)`, `border_set(3XCURSES)`, `libcurses(3XCURSES)`,
`attributes(5)`, `standards(5)`

NAME border_set, box_set, wborder_set – use complex characters (and renditions) to draw borders

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

int border_set(const cchar_t *ls, const cchar_t *rs, const cchar_t
  *ts, const cchar_t *bs, const cchar_t *tl, const cchar_t *tr,
  const cchar_t *bl, const cchar_t *br);

int wborder_set(WINDOW *win, const cchar_t *ls, const cchar_t *rs,
  const cchar_t *ts, const cchar_t *bs, const cchar_t *tl, const
  cchar_t *tr, const cchar_t *bl, const cchar_t *br);

int box_set(WINDOW *win, const cchar_t *verch, const cchar_t
  *horch);
```

DESCRIPTION The border_set () and wborder_set () functions draw a border around the specified window. All parameters must be spacing complex characters with renditions. A parameter which is a null pointer is replaced by the default character.

Constant Values for Borders

| Constant Values for Borders | | |
|-----------------------------|------------------|-------------------|
| Parameter | Default Constant | Default Character |
| <i>verch</i> | WACS_VLINE | |
| <i>horch</i> | WACS_HLINE | - |
| <i>ls</i> | WACS_VLINE | |
| <i>rs</i> | WACS_VLINE | |
| <i>ts</i> | WACS_HLINE | - |
| <i>bs</i> | WACS_HLINE | - |
| <i>bl</i> | WACS_BLCORNER | + |
| <i>br</i> | WACS_BRCORNER | + |
| <i>tl</i> | WACS_ULCORNER | + |
| <i>tr</i> | WACS_URCORNER | + |

The call

wborder_set(3XCURSES)

`box_set(win, verch, horch)` is a short form for

```
wborder(win, verch, verch, horch, horch, NULL, NULL, NULL, NULL)
```

When the window is boxed, the bottom and top rows and right and left columns are unavailable for text.

PARAMETERS

ls Is the character and rendition used for the left side of the border.

rs Is the character and rendition used for the right side of the border.

ts Is the character and rendition used for the top of the border.

bs Is the character and rendition used for the bottom of the border.

tl Is the character and rendition used for the top-left corner of the border.

tr Is the character and rendition used for the top-right corner of the border.

bl Is the character and rendition used for the bottom-left corner of the border.

br Is the character and rendition used for the bottom-right corner of the border.

win Is the pointer to the window in which the border or box is to be drawn.

verch Is the character and rendition used for the left and right columns of the box.

horch Is the character and rendition used for the top and bottom rows of the box.

RETURN VALUES

On success, these functions return OK. Otherwise, they return ERR.

ERRORS

None.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

wborder_set(3XCURSES)

SEE ALSO add_wch(3XCURSES), addch(3XCURSES), attr_get(3XCURSES),
attroff(3XCURSES), border(3XCURSES), libcurses(3XCURSES),
attributes(5), standards(5)

wchgat(3XCURSES)

| | | | | | | | | | | | | | | | |
|----------------------|--|----------|---|-------------|--|--------------|---|-------------|---|----------|---|----------|---|------------|---|
| NAME | chgat, mvchgat, mvwchgat, wchgat – change the rendition of characters in a window | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int chgat(int <i>n</i>, attr_t <i>attr</i>, short <i>color</i>, const void *<i>opts</i>); int mvchgat(int <i>y</i>, int <i>x</i>, int <i>n</i>, attr_t <i>attr</i>, short <i>color</i>, const void *<i>opts</i>); int mvwchgat(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>, int <i>n</i>, attr_t <i>attr</i>, short <i>color</i>, const void *<i>opts</i>); int wchgat(WINDOW *<i>win</i>, int <i>n</i>, attr_t <i>attr</i>, short <i>color</i>, const void *<i>opts</i>);</pre> | | | | | | | | | | | | | | |
| DESCRIPTION | <p>These functions change the renditions of the next <i>n</i> characters in the current or specified window (or of the remaining characters on the current or specified line, if <i>n</i> is -1), beginning at the current or specified cursor position. The attributes and colors are specified by <i>attr</i> and <i>color</i> as for <code>setcchar(3XCURSES)</code>.</p> <p>These function neither update the cursor nor perform wrapping.</p> <p>A value of <i>n</i> that is greater than the remaining characters on a line is not an error.</p> <p>The <i>opts</i> argument is reserved for definition in a future release. Currently, the application must provide a null pointer for <i>opts</i>.</p> | | | | | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>n</i></td><td>Is the number of characters whose rendition is to be changed.</td></tr><tr><td><i>attr</i></td><td>Is the set of attributes to be assigned to the characters.</td></tr><tr><td><i>color</i></td><td>Is the new color pair to be assigned to the characters.</td></tr><tr><td><i>opts</i></td><td>Is reserved for future use. Currently, this must be a null pointer.</td></tr><tr><td><i>y</i></td><td>Is the y (row) coordinate of the starting position in the window.</td></tr><tr><td><i>x</i></td><td>Is the x (column) coordinate of the starting position in the window. changed in the window.</td></tr><tr><td><i>win</i></td><td>Is a pointer to the window in which the rendition of characters is to be changed.</td></tr></table> | <i>n</i> | Is the number of characters whose rendition is to be changed. | <i>attr</i> | Is the set of attributes to be assigned to the characters. | <i>color</i> | Is the new color pair to be assigned to the characters. | <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | <i>y</i> | Is the y (row) coordinate of the starting position in the window. | <i>x</i> | Is the x (column) coordinate of the starting position in the window. changed in the window. | <i>win</i> | Is a pointer to the window in which the rendition of characters is to be changed. |
| <i>n</i> | Is the number of characters whose rendition is to be changed. | | | | | | | | | | | | | | |
| <i>attr</i> | Is the set of attributes to be assigned to the characters. | | | | | | | | | | | | | | |
| <i>color</i> | Is the new color pair to be assigned to the characters. | | | | | | | | | | | | | | |
| <i>opts</i> | Is reserved for future use. Currently, this must be a null pointer. | | | | | | | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of the starting position in the window. | | | | | | | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of the starting position in the window. changed in the window. | | | | | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the rendition of characters is to be changed. | | | | | | | | | | | | | | |
| RETURN VALUES | Upon successful completion, these functions returned OK. Otherwise, they return ERR. | | | | | | | | | | | | | | |
| ERRORS | No errors are defined. | | | | | | | | | | | | | | |

wchgat(3XCURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `bkgnd(3XCURSES)`, `libcurses(3XCURSES)`, `setcchar(3XCURSES)`, `attributes(5)`, `standards(5)`

wclear(3CURSES)

| NAME | <code> curs_clear, erase, werase, clear, wclear, clrtoeol, wclrtoeol, clrtoeol, wclrtoeol</code> – clear all or part of a curses window | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file ... -lcurses [library ...] #include <curses.h> int erase(void); int werase(WINDOW *win); int clear(void); int wclear(WINDOW *win); int clrtoeol(void); int wclrtoeol(WINDOW *win); int clrtoeol(void); int wclrtoeol(WINDOW *win);</pre> | | | | |
| DESCRIPTION | <p>The <code>erase()</code> and <code>werase()</code> routines copy blanks to every position in the window.</p> <p>The <code>clear()</code> and <code>wclear()</code> routines are like <code>erase()</code> and <code>werase()</code>, but they also call <code>clearok()</code>, so that the screen is cleared completely on the next call to <code>wrefresh()</code> for that window and repainted from scratch.</p> <p>The <code>clrtoeol()</code> and <code>wclrtoeol()</code> routines erase all lines below the cursor in the window. Also, the current line to the right of the cursor, inclusive, is erased.</p> <p>The <code>clrtoeol()</code> and <code>wclrtoeol()</code> routines erase the current line to the right of the cursor, inclusive.</p> | | | | |
| RETURN VALUES | All routines return the integer <code>OK</code> , or a non-negative integer if <code>immedok()</code> is set. See <code>curls_ouptops(3CURSES)</code> . | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curls_ouptops(3CURSES)</code> , <code>curls_refresh(3CURSES)</code> , <code>curls(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that <code>erase()</code>, <code>werase()</code>, <code>clear()</code>, <code>wclear()</code>, <code>clrtoeol()</code>, and <code>clrtoeol()</code> may be macros.</p> | | | | |

| NAME | clear, erase, wclear, werase – clear a window | | | | | | |
|---------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int clear(void); int erase(void); int wclear(WINDOW *win); int werase(WINDOW *win);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>clear()</code> and <code>erase()</code> functions clear <code>stdscr</code>, destroying its previous contents. The <code>wclear()</code> and <code>werase()</code> functions perform the same action, but clear the window specified by <code>win</code> instead of <code>stdscr</code>.</p> <p>The <code>clear()</code> and <code>wclear()</code> functions also call the <code>clearok()</code> function. This function clears and redraws the entire screen on the next call to <code>refresh(3XCURSES)</code> or <code>wrefresh(3XCURSES)</code> for the window.</p> <p>The current background character (and attributes) is used to clear the screen.</p> | | | | | | |
| PARAMETERS | <p><i>win</i> Is a pointer to the window that is to be cleared.</p> | | | | | | |
| ERRORS | <p>OK Successful completion.</p> <p>ERR An error occurred.</p> | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <p><code>bkgdset(3XCURSES)</code>, <code>clearok(3XCURSES)</code>, <code>clrtoeol(3XCURSES)</code>, <code>clrtoeol(3XCURSES)</code>, <code>doupdate(3XCURSES)</code>, <code>libcurses(3XCURSES)</code>, <code>refresh(3XCURSES)</code>, <code>wrefresh(3XCURSES)</code>, <code>attributes(5)</code>, <code>standards(5)</code></p> | | | | | | |

wclrtoobot(3CURSES)

NAME | curs_clear, erase, werase, clear, wclear, clrtoobot, wclrtoobot, clrtoeol, wclrtoeol – clear all or part of a curses window

SYNOPSIS |

```
cc [ flag... ] file ... -lcurses [ library ... ]
#include <curses.h>

int erase(void);
int werase(WINDOW *win);
int clear(void);
int wclear(WINDOW *win);
int clrtoobot(void);
int wclrtoobot(WINDOW *win);
int clrtoeol(void);
int wclrtoeol(WINDOW *win);
```

DESCRIPTION | The erase() and werase() routines copy blanks to every position in the window. The clear() and wclear() routines are like erase() and werase(), but they also call clearok(), so that the screen is cleared completely on the next call to wrefresh() for that window and repainted from scratch. The clrtoobot() and wclrtoobot() routines erase all lines below the cursor in the window. Also, the current line to the right of the cursor, inclusive, is erased. The clrtoeol() and wclrtoeol() routines erase the current line to the right of the cursor, inclusive.

RETURN VALUES | All routines return the integer OK, or a non-negative integer if immedok() is set. See curs_ouptops(3CURSES).

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | curs_ouptops(3CURSES), curs_refresh(3CURSES), curses(3CURSES), attributes(5)

NOTES | The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>. Note that erase(), werase(), clear(), wclear(), clrtoobot(), and clrtoeol() may be macros.

| NAME | clrtoobot, wclrtoobot – clear to the end of a window | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int clrtoobot(void); int wclrtoobot(WINDOW *win);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>clrtoobot()</code> function clears all characters in the <code>stdscr</code> window from the cursor to the end of the window. The <code>wclrtoobot()</code> function performs the same action in the window specified by <i>win</i> instead of in <code>stdscr</code>. The current background character (and rendition) is used to clear the screen.</p> <p>If the clearing action results in clearing only a portion of a multicolumn character, background characters are displayed in place of the remaining portion.</p> | | | | | | |
| PARAMETERS | <p><i>win</i> Is a pointer to the window that is to be cleared.</p> | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>bkgdset(3XCURSES)</code> , <code>clear(3XCURSES)</code> , <code>clearok(3XCURSES)</code> , <code>crltoeol(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

wclrtoeol(3CURSES)

NAME | curs_clear, erase, werase, clear, wclear, clrtoeol, wclrtoeol – clear all or part of a curses window

SYNOPSIS | `cc [flag...] file ... -lcurses [library ...]`
`#include <curses.h>`

```
int erase(void);
int werase(WINDOW *win);
int clear(void);
int wclear(WINDOW *win);
int clrtoeol(void);
int wclrtoeol(WINDOW *win);
int clrtoeol(void);
int wclrtoeol(WINDOW *win);
```

DESCRIPTION | The erase() and werase() routines copy blanks to every position in the window.

The clear() and wclear() routines are like erase() and werase(), but they also call clearok(), so that the screen is cleared completely on the next call to wrefresh() for that window and repainted from scratch.

The clrtoeol() and wclrtoeol() routines erase all lines below the cursor in the window. Also, the current line to the right of the cursor, inclusive, is erased.

The clrtoeol() and wclrtoeol() routines erase the current line to the right of the cursor, inclusive.

RETURN VALUES | All routines return the integer OK, or a non-negative integer if immedok() is set. See curs_ouptops(3CURSES).

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | curs_ouptops(3CURSES), curs_refresh(3CURSES), curses(3CURSES), attributes(5)

NOTES | The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.

Note that erase(), werase(), clear(), wclear(), clrtoeol(), and clrtoeol() may be macros.

| NAME | clrtoeol, wclrtoeol – clear to the end of a line | | | | | | |
|----------------------|--|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int clrtoeol(void); int wclrtoeol(WINDOW *win);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>clrtoeol()</code> function clears the current line from the cursor to the right margin in the <code>stdscr</code> window. The <code>wclrtoeol()</code> function performs the same action, but in the window specified by <i>win</i> instead of <code>stdscr</code>. The current background character (and rendition) is used to clear the screen.</p> <p>If the clearing action results in clearing only a portion of a multicolumn character, background characters are displayed in place of the remaining portion.</p> | | | | | | |
| PARAMETERS | <p><i>win</i> Is a pointer to the window in which to clear to the end of the line.</p> | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return FALSE. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>bkgdset(3XCURSES)</code> , <code>clear(3XCURSES)</code> , <code>clearok(3XCURSES)</code> , <code>clrtoeol(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

wcolor_set(3XCURSES)

| | | | | | | | | | |
|--------------------|--|--------------------|--|--------------------|---------------------------------------|-------------------|-----------------------------|------------------|---|
| NAME | <code>attr_get</code> , <code>attr_off</code> , <code>attr_on</code> , <code>attr_set</code> , <code>color_set</code> , <code>wattr_get</code> , <code>wattr_off</code> , <code>wattr_on</code> , <code>wattr_set</code> , <code>wcolor_set</code> – control window attributes | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int attr_get(attr_t *attrs, short *color, void *opts); int attr_off(attr_t attrs, void *opts); int attr_on(attr_t attrs, void *opts); int attr_set(attr_t attrs, short color, void *opts); int color_set(short *color, void *opts); int wattr_get(WINDOW *win, attr_t attrs, short *color, void *opts); int wattr_off(WINDOW *win, attr_t attrs, void *opts); int wattr_on(WINDOW *win, attr_t attrs, void *opts); int wattr_set(WINDOW *win, attr_t attrs, short color, void *opts); int wcolor_set(WINDOW *win, short color, void *opts);</pre> | | | | | | | | |
| DESCRIPTION | <p>The <code>attr_get()</code> function retrieves the current rendition of <code>stdscr</code>. The <code>wattr_get()</code> function retrieves the current rendition of window <code>win</code>. If <code>attrs</code> or <code>color</code> is a null pointer, no information is retrieved.</p> <p>The <code>attr_off()</code> and <code>attr_on()</code> functions unset and set, respectively, the specified window attributes of <code>stdscr</code>. These functions only affect the attributes specified; attributes that existed before the call are retained.</p> <p>The <code>wattr_off()</code> and <code>wattr_on()</code> functions unset or set the specified attributes for window <code>win</code>.</p> <p>The <code>attr_set()</code> and <code>wattr_set()</code> functions change the rendition of <code>stdscr</code> and <code>win</code>; the old values are not retained.</p> <p>The <code>color_set()</code> and <code>wcolor_set()</code> functions set the window color of <code>stdscr</code> and <code>win</code> to <code>color</code>.</p> <p>The attributes and color pairs that can be used are specified in the <code>Attributes</code>, <code>Color Pairs</code>, and <code>Renditions</code> section of the <code>curses(3XCURSES)</code> man page.</p> | | | | | | | | |
| PARAMETERS | <table><tr><td><code>attrs</code></td><td>Is a pointer to the foreground window attributes to be set or unset.</td></tr><tr><td><code>color</code></td><td>Is a pointer to a color pair number .</td></tr><tr><td><code>opts</code></td><td>Is reserved for future use.</td></tr><tr><td><code>win</code></td><td>Is a pointer to the window in which attribute changes are to be made.</td></tr></table> | <code>attrs</code> | Is a pointer to the foreground window attributes to be set or unset. | <code>color</code> | Is a pointer to a color pair number . | <code>opts</code> | Is reserved for future use. | <code>win</code> | Is a pointer to the window in which attribute changes are to be made. |
| <code>attrs</code> | Is a pointer to the foreground window attributes to be set or unset. | | | | | | | | |
| <code>color</code> | Is a pointer to a color pair number . | | | | | | | | |
| <code>opts</code> | Is reserved for future use. | | | | | | | | |
| <code>win</code> | Is a pointer to the window in which attribute changes are to be made. | | | | | | | | |

wcolor_set(3XCURSES)

RETURN VALUES These functions always return OK.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO add_wch(3XCURSES), addnwstr(3XCURSES), attroff(3XCURSES), bkgrndset(3XCURSES), curses(3XCURSES), init_color(3XCURSES), libcurses(3XCURSES), start_color(3XCURSES), attributes(5), standards(5)

wcursyncup(3CURSES)

| | |
|--------------------|--|
| NAME | <code>curs_window</code> , <code>newwin</code> , <code>delwin</code> , <code>mvwin</code> , <code>subwin</code> , <code>derwin</code> , <code>mvderwin</code> , <code>dupwin</code> , <code>wsyncup</code> , <code>syncok</code> , <code>wcursyncup</code> , <code>wsyncdown</code> – create curses windows |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> WINDOW *newwin(int nlines, int ncols, int begin_y, int begin_x); int delwin(WINDOW *win); int mvwin(WINDOW *win, int y, int x); WINDOW *subwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); WINDOW *derwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); int mvderwin(WINDOW *win, int par_y, int par_x); WINDOW *dupwin(WINDOW *win); void wsyncup(WINDOW *win); int syncok(WINDOW *win, bool bf); void wcursyncup(WINDOW *win); void wsyncdown(WINDOW *win);</pre> |
| DESCRIPTION | <p>The <code>newwin()</code> routine creates and returns a pointer to a new window with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. The upper left-hand corner of the window is at line <i>begin_y</i>, column <i>begin_x</i>. If either <i>nlines</i> or <i>ncols</i> is zero, they default to <code>LINES</code> — <i>begin_y</i> and <code>COLS</code> — <i>begin_x</i>. A new full-screen window is created by calling <code>newwin(0, 0, 0, 0)</code>.</p> <p>The <code>delwin()</code> routine deletes the named window, freeing all memory associated with it. Subwindows must be deleted before the main window can be deleted.</p> <p>The <code>mvwin()</code> routine moves the window so that the upper left-hand corner is at position (<i>x</i>, <i>y</i>). If the move would cause the window to be off the screen, it is an error and the window is not moved. Moving subwindows is allowed, but should be avoided.</p> <p>The <code>subwin()</code> routine creates and returns a pointer to a new window with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. The window is at position (<i>begin_y</i>, <i>begin_x</i>) on the screen. (This position is relative to the screen, and not to the window <i>orig</i>.) The window is made in the middle of the window <i>orig</i>, so that changes made to one window will affect both windows. The subwindow shares memory with the window <i>orig</i>. When using this routine, it is necessary to call <code>touchwin()</code> or <code>touchline()</code> on <i>orig</i> before calling <code>wrefresh()</code> on the subwindow.</p> <p>The <code>derwin()</code> routine is the same as <code>subwin()</code>, except that <i>begin_y</i> and <i>begin_x</i> are relative to the origin of the window <i>orig</i> rather than the screen. There is no difference between the subwindows and the derived windows.</p> |

wcursyncup(3CURSES)

The `mvderwin()` routine moves a derived window (or subwindow) inside its parent window. The screen-relative parameters of the window are not changed. This routine is used to display different parts of the parent window at the same physical position on the screen.

The `dupwin()` routine creates an exact duplicate of the window *win*.

Each `curses` window maintains two data structures: the character image structure and the status structure. The character image structure is shared among all windows in the window hierarchy (that is, the window with all subwindows). The status structure, which contains information about individual line changes in the window, is private to each window. The routine `wrefresh()` uses the status data structure when performing screen updating. Since status structures are not shared, changes made to one window in the hierarchy may not be properly reflected on the screen.

The routine `wsyncup()` causes the changes in the status structure of a window to be reflected in the status structures of its ancestors. If `syncok()` is called with second argument `TRUE` then `wsyncup()` is called automatically whenever there is a change in the window.

The routine `wcursyncup()` updates the current cursor position of all the ancestors of the window to reflect the current cursor position of the window.

The routine `wsyncdown()` updates the status structure of the window to reflect the changes in the status structures of its ancestors. Applications seldom call this routine because it is called automatically by `wrefresh()`.

RETURN VALUES

Routines that return an integer return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`delwin()` returns the integer `ERR` upon failure and `OK` upon successful completion.

Routines that return pointers return `NULL` on error.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO

`curs_refresh(3CURSES)`, `curs_touch(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

If many small changes are made to the window, the `wsyncup()` option could degrade performance.

wcursyncup(3CURSES)

Note that `syncok()` may be a macro.

NAME syncok, wcursyncup, wsyncdown, wsyncup – synchronize window with its parents or children

SYNOPSIS

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

int syncok(WINDOW *win, bool bf);

void wcursyncup(WINDOW *win);

void wsyncdown(WINDOW *win);

void wsyncup(WINDOW *win);
```

PARAMETERS

win Is a pointer to a window.

bf Is a Boolean expression.

DESCRIPTION

The syncok() function uses the value of *bf* to determine whether or not the window *win*'s ancestors are implicitly touched whenever there is a change to *win*. If *bf* is TRUE, this touching occurs. If *bf* is FALSE, it does not occur. The initial value for *bf* is FALSE.

The wcursyncup() function moves the cursor in *win*'s ancestors to match its position in *win*.

The wsyncdown() function touches *win* if any of its ancestors have been touched.

The wsyncup() function touches all ancestors of *win*.

RETURN VALUES

On success, the syncok() function returns OK. Otherwise, it returns ERR.

The other functions do not return a value.

ERRORS

None.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO derwin(3XCURSES), doupdate(3XCURSES), is_linetouched(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

wdelch(3CURSES)

NAME curs_delch, delch, wdelch, mvdelch, mvwdelch – delete character under cursor in a curses window

SYNOPSIS

```
cc [ flag ... ] file ... -lcurses [ library ... ]
#include <curses.h>

int delch(void);
int wdelch(WINDOW *win);
int mvdelch(int y, int x);
int mvwdelch(WINDOW *win, int y, int x);
```

DESCRIPTION With these routines the character under the cursor in the window is deleted; all characters to the right of the cursor on the same line are moved to the left one position and the last character on the line is filled with a blank. The cursor position does not change (after moving to *y, x*, if specified). This does not imply use of the hardware delete character feature.

RETURN VALUES All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curses(3CURSES), attributes(5)

NOTES The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `delch()`, `mvdelch()`, and `mvwdelch()` may be macros.

| NAME | delch, mvdelch, mvwdelch, wdelch – remove a character | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int delch(void); int mvdelch(int <i>y</i>, int <i>x</i>); int mvwdelch(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>); int wdelch(WINDOW *<i>win</i>);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>delch()</code> and <code>wdelch()</code> functions delete the character at the current cursor position from <code>stdscr</code> and <i>win</i>, respectively. All remaining characters after cursor through to the end of the line are shifted one character towards the start of the line. The last character on the line becomes a space; characters on other lines are not affected.</p> <p>The <code>mvdelch()</code> and <code>mvwdelch()</code> functions delete the character at the position specified by the <i>x</i> and <i>y</i> parameters; the former deletes the character from <code>stdscr</code>; the latter from <i>win</i>.</p> | | | | | | |
| PARAMETERS | <p><i>y</i> Is the <i>y</i> (row) coordinate of the position of the character to be removed.</p> <p><i>x</i> Is the <i>x</i> (column) coordinate of the position of the character to be removed.</p> <p><i>win</i> Is a pointer to the window containing the character to be removed.</p> | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>bkgdset(3XCURSES)</code> , <code>insch(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code> | | | | | | |

wdeleteln(3CURSES)

| NAME | <code> curs_deleteln, deleteln, wdeleteln, insdelln, winsdelln, insertln, winsertln</code> – delete and insert lines in a curses window | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int deleteln(void); int wdeleteln(WINDOW *win); int insdelln(int <i>n</i>); int winsdelln(WINDOW *win, int <i>n</i>); int insertln(void); int winsertln(WINDOW *win);</pre> | | | | |
| DESCRIPTION | <p>With the <code>deleteln()</code> and <code>wdeleteln()</code> routines, the line under the cursor in the window is deleted; all lines below the current line are moved up one line. The bottom line of the window is cleared. The cursor position does not change. This does not imply use of a hardware delete line feature.</p> <p>With the <code>insdelln()</code> and <code>winsdelln()</code> routines, for positive <i>n</i>, insert <i>n</i> lines into the specified window above the current line. The <i>n</i> bottom lines are lost. For negative <i>n</i>, delete <i>n</i> lines (starting with the one under the cursor), and move the remaining lines up. The bottom <i>n</i> lines are cleared. The current cursor position remains the same.</p> <p>With the <code>insertln()</code> and <code>winsertln()</code> routines, a blank line is inserted above the current line and the bottom line is lost. This does not imply use of a hardware insert line feature.</p> | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . Note that all but <code>winsdelln()</code> may be macros. | | | | |

NAME deleteln, wdeleteln – remove a line

SYNOPSIS `cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \`
`-R /usr/xpg4/lib -lcurses [library...]`

`c89 [flag...] file... -lcurses [library...]`

```
#include <curses.h>
int deleteln(void);
int wdeleteln(WINDOW *win);
```

DESCRIPTION The `deleteln()` and `wdeleteln()` functions delete the line containing the cursor from `stdscr` and `win`, respectively. All lines below the one deleted are moved up one line. The last line of the window becomes blank. The position of the cursor is unchanged.

PARAMETERS *win* Is a pointer to the window from which the line is removed.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `bkgdset(3XCURSES)`, `insdelln(3XCURSES)`, `insertln(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

wechochar(3CURSES)

| | |
|----------------------|--|
| NAME | <code>curs_addch</code> , <code>addch</code> , <code>waddch</code> , <code>mvaddch</code> , <code>mvwaddch</code> , <code>echochar</code> , <code>wechochar</code> – add a character (with attributes) to a curses window and advance cursor |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int addch(chtype <i>ch</i>); int waddch(WINDOW *<i>win</i>, chtype <i>ch</i>); int mvaddch(int <i>y</i>, int <i>x</i>, chtype <i>ch</i>); int mvwaddch(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>, chtype <i>ch</i>); int echochar(chtype <i>ch</i>); int wechochar(WINDOW *<i>win</i>, chtype <i>ch</i>);</pre> |
| DESCRIPTION | <p>With the <code>addch()</code>, <code>waddch()</code>, <code>mvaddch()</code>, and <code>mvwaddch()</code> routines, the character <i>ch</i> is put into the window at the current cursor position of the window and the position of the window cursor is advanced. Its function is similar to that of <code>putchar()</code>. At the right margin, an automatic newline is performed. At the bottom of the scrolling region, if <code>scrollok()</code> is enabled, the scrolling region is scrolled up one line.</p> <p>If <i>ch</i> is a tab, newline, or backspace, the cursor is moved appropriately within the window. A newline also does a <code>clrtoeol()</code> before moving. Tabs are considered to be at every eighth column. If <i>ch</i> is another control character, it is drawn in the <code>^X</code> notation. Calling <code>winch()</code> after adding a control character does not return the control character, but instead returns the representation of the control character. See <code>curs_inch(3CURSES)</code>.</p> <p>Video attributes can be combined with a character by OR-ing them into the parameter. This results in these attributes also being set. (The intent here is that text, including attributes, can be copied from one place to another using <code>inch()</code> and <code>addch()</code>.) (see <code>standout()</code>, predefined video attribute constants, on the <code>curs_attr(3CURSES)</code> page).</p> <p>The <code>echochar()</code> and <code>wechochar()</code> routines are functionally equivalent to a call to <code>addch()</code> followed by a call to <code>refresh()</code>, or a call to <code>waddch</code> followed by a call to <code>wrefresh()</code>. The knowledge that only a single character is being output is taken into consideration and, for non-control characters, a considerable performance gain might be seen by using these routines instead of their equivalents.</p> |
| Line Graphics | <p>The following variables may be used to add line drawing characters to the screen with routines of the <code>addch()</code> family. When variables are defined for the terminal, the <code>A_ALTCHARSET</code> bit is turned on (see <code>curs_attr(3CURSES)</code>). Otherwise, the default character listed below is stored in the variable. The names chosen are consistent with the VT100 nomenclature.</p> |

| Name | Default | Glyph Description |
|--------------|---------|-------------------------|
| ACS_ULCORNER | + | upper left-hand corner |
| ACS_LLCORNER | + | lower left-hand corner |
| ACS_URCORNER | + | upper right-hand corner |
| ACS_LRCORNER | + | lower right-hand corner |
| ACS_RTEE | + | right tee |
| ACS_LTEE | + | left tee |
| ACS_BTEE | + | bottom tee |
| ACS_TTEE | + | top tee |
| ACS_HLINE | - | horizontal line |
| ACS_VLINE | | vertical line |
| ACS_PLUS | + | plus |
| ACS_S1 | - | scan line 1 |
| ACS_S9 | - | scan line 9 |
| ACS_DIAMOND | + | diamond |
| ACS_CKBOARD | : | checker board (stipple) |
| ACS_DEGREE | ' | degree symbol |
| ACS_PLMINUS | # | plus/minus |
| ACS_BULLET | o | bullet |
| ACS_LARROW | < | arrow pointing left |
| ACS_RARROW | > | arrow pointing right |
| ACS_DARROW | v | arrow pointing down |
| ACS_UARROW | ^ | arrow pointing up |
| ACS_BOARD | # | board of squares |
| ACS_LANTERN | # | lantern symbol |
| ACS_BLOCK | # | solid square block |

RETURN VALUES

All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

wechochar(3CURSES)

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_attr(3CURSES), curs_clear(3CURSES), curs_inch(3CURSES), curs_outopts(3CURSES), curs_refresh(3CURSES), curses(3CURSES), putc(3C), attributes(5)

NOTES The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.

Note that addch(), mvaddch(), mvwaddch(), and echochar() may be macros.

| NAME | echochar, wechochar – add a single-byte character and refresh window | | | | | | |
|----------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int echochar(const chtype <i>ch</i>); int wechochar(WINDOW *<i>win</i>, const chtype <i>ch</i>);</pre> | | | | | | |
| DESCRIPTION | The echochar() function produces the same effect as calling addch(3XCURSES) and then refresh(3XCURSES). The wechochar() function produces the same effect as calling waddch(3XCURSES) and then wrefresh(3XCURSES). | | | | | | |
| PARAMETERS | <p><i>ch</i> Is a pointer to the character to be written to the window.</p> <p><i>win</i> Is a pointer to the window in which the character is to be added.</p> | | | | | | |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. | | | | | | |
| ERRORS | None. | | | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | addch(3XCURSES), doupdate(3XCURSES), echo_wchar(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5) | | | | | | |

wechownchar(3CURSES)

| | |
|----------------------|---|
| NAME | <code> curs_ addwch , addwch , waddwch , mvaddwch , mvwaddwch , echowchar , wechownchar </code> – add a <code>wchar_t</code> character (with attributes) to a curses window and advance cursor |
| SYNOPSIS | <pre>cc [flag...] file... -lcurses [library...] #include<curses.h> int addwch (chtype wch) ; int waddwch (WINDOW *win , chtype wch) ; int mvaddwch (int y , int x , chtype wch) ; int mvwaddwch (WINDOW *win , int y , int x , chtype wch) ; int echowchar (chtype wch) ; int wechownchar (WINDOW *win , chtype wch) ;</pre> |
| DESCRIPTION | <p>The <code>addwch()</code>, <code>waddwch()</code>, <code>mvaddwch()</code>, and <code>mvwaddwch()</code> routines put the character <code>wch</code>, holding a <code>wchar_t</code> character, into the window at the current cursor position of the window and advance the position of the window cursor. Their function is similar to that of <code>putwchar(3C)</code> in the C multibyte library. At the right margin, an automatic newline is performed. At the bottom of the scrolling region, if <code>scrollok</code> is enabled, the scrolling region is scrolled up one line.</p> <p>If <code>wch</code> is a tab, newline, or backspace, the cursor is moved appropriately within the window. A newline also does a <code>clrtoeol(3CURSES)</code> before moving. Tabs are considered to be at every eighth column. If <code>wch</code> is another control character, it is drawn in the <code>^X</code> notation. Calling <code>winwch(3CURSES)</code> after adding a control character does not return the control character, but instead returns the representation of the control character.</p> <p>Video attributes can be combined with a <code>wchar_t</code> character by OR-ing them into the parameter. This results in these attributes also being set. (The intent here is that text, including attributes, can be copied from one place to another using <code>inwch()</code> and <code>addwch()</code>.) See <code>standout(3CURSES)</code>, predefined video attribute constants.</p> <p>The <code>echowchar()</code> and <code>wechownchar()</code> routines are functionally equivalent to a call to <code>addwch()</code> followed by a call to <code>refresh(3CURSES)</code>, or a call to <code>waddwch()</code> followed by a call to <code>wrefresh(3CURSES)</code>. The knowledge that only a single character is being output is taken into consideration and, for non-control characters, a considerable performance gain might be seen by using these routines instead of their equivalents.</p> |
| Line Graphics | The following variables may be used to add line drawing characters to the screen with routines of the <code>addwch()</code> family. When variables are defined for the terminal, the <code>A_ALTCHARSET</code> bit is turned on. (See <code> curs_attr(3CURSES)</code>). Otherwise, the default character listed below is stored in the variable. The names chosen are consistent with the VT100 nomenclature. |

wechowchar(3CURSES)

| Name | Default | Glyph Description |
|--------------|---------|-------------------------|
| ACS_ULCORNER | + | upper left-hand corner |
| ACS_LLCORNER | + | lower left-hand corner |
| ACS_URCORNER | + | upper right-hand corner |
| ACS_LRCORNER | + | lower right-hand corner |
| ACS_RTEE | + | right tee |
| ACS_LTEE | + | left tee |
| ACS_BTEE | + | bottom tee |
| ACS_TTEE | + | top tee |
| ACS_HLINE | - | horizontal line |
| ACS_VLINE | | vertical line |
| ACS_PLUS | + | plus |
| ACS_S1 | - | scan line 1 |
| ACS_S9 | - | scan line 9 |
| ACS_DIAMOND | + | diamond |
| ACS_CKBOARD | : | checker board (stipple) |
| ACS_DEGREE | ' | degree symbol |
| ACS_PLMINUS | # | plus/minus |
| ACS_BULLET | o | bullet |
| ACS_LARROW | < | arrow pointing left |
| ACS_RARRROW | > | arrow pointing right |
| ACS_DARROW | v | arrow pointing down |
| ACS_UARROW | ^ | arrow pointing up |
| ACS_BOARD | # | board of squares |
| ACS_LANTERN | # | lantern symbol |
| ACS_BLOCK | # | solid square block |

RETURN VALUE All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion, unless otherwise noted in the preceding routine descriptions.

wechownchar(3CURSES)

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | `putwchar(3C)`, `clrtoeol(3CURSES)`, `curses(3CURSES)`, `curl_attr(3CURSES)`, `curl_inwch(3CURSES)`, `curl_outopts(3CURSES)`, `refresh(3CURSES)`, `standout(3CURSES)`, `winwch(3CURSES)`, `wrefresh(3CURSES)`, `attributes(5)`

NOTES | The header file `<curses.h>` automatically includes the header files `<stdio.h>`, `<unctrl.h>` and `<widec.h>`.

Note that `addwch()`, `mvaddwch()`, `mvwaddwch()`, and `echowchar()` may be macros.

None of these routines can use the color attribute in `chtype`.

| | |
|----------------------|--|
| NAME | echo_wchar, wecho_wchar – add a complex character and refresh window |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int echo_wchar(const cchar_t *wch); int wecho_wchar(WINDOW *win, const cchar_t *wch);</pre> |
| DESCRIPTION | The echo_wchar() function produces the same effect as calling add_wch(3XCURSES) and then refresh(3XCURSES). The wecho_wchar() function produces the same effect as calling wadd_wch(3XCURSES) and then wrefresh(3XCURSES). |
| PARAMETERS | <p><i>wch</i> Is a pointer to the complex character to be written to the window.</p> <p><i>win</i> Is a pointer to the window in which the character is to be added.</p> |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. |
| ERRORS | None. |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: |

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO add_wch(3XCURSES), doupdate(3XCURSES), echochar(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

werase(3CURSES)

NAME | curs_clear, erase, werase, clear, wclear, clrtoobot, wclrtoobot, clrtoeol, wclrtoeol – clear all or part of a curses window

SYNOPSIS |

```
cc [ flag... ] file ... -lcurses [ library ... ]
#include <curses.h>

int erase(void);
int werase(WINDOW *win);
int clear(void);
int wclear(WINDOW *win);
int clrtoobot(void);
int wclrtoobot(WINDOW *win);
int clrtoeol(void);
int wclrtoeol(WINDOW *win);
```

DESCRIPTION | The erase() and werase() routines copy blanks to every position in the window. The clear() and wclear() routines are like erase() and werase(), but they also call clearok(), so that the screen is cleared completely on the next call to wrefresh() for that window and repainted from scratch. The clrtoobot() and wclrtoobot() routines erase all lines below the cursor in the window. Also, the current line to the right of the cursor, inclusive, is erased. The clrtoeol() and wclrtoeol() routines erase the current line to the right of the cursor, inclusive.

RETURN VALUES | All routines return the integer OK, or a non-negative integer if immedok() is set. See curs_ouptos(3CURSES).

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | curs_ouptos(3CURSES), curs_refresh(3CURSES), curses(3CURSES), attributes(5)

NOTES | The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>. Note that erase(), werase(), clear(), wclear(), clrtoobot(), and clrtoeol() may be macros.

| NAME | clear, erase, wclear, werase – clear a window | | | | | | |
|---------------------|---|----------------|-----------------|---------------------|----------|----------|--------|
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int clear(void); int erase(void); int wclear(WINDOW *win); int werase(WINDOW *win);</pre> | | | | | | |
| DESCRIPTION | <p>The <code>clear()</code> and <code>erase()</code> functions clear <code>stdscr</code>, destroying its previous contents. The <code>wclear()</code> and <code>werase()</code> functions perform the same action, but clear the window specified by <code>win</code> instead of <code>stdscr</code>.</p> <p>The <code>clear()</code> and <code>wclear()</code> functions also call the <code>clearok()</code> function. This function clears and redraws the entire screen on the next call to <code>refresh(3XCURSES)</code> or <code>wrefresh(3XCURSES)</code> for the window.</p> <p>The current background character (and attributes) is used to clear the screen.</p> | | | | | | |
| PARAMETERS | <p><i>win</i> Is a pointer to the window that is to be cleared.</p> | | | | | | |
| ERRORS | <p>OK Successful completion.</p> <p>ERR An error occurred.</p> | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| Interface Stability | Standard | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <p><code>bkgdset(3XCURSES)</code>, <code>clearok(3XCURSES)</code>, <code>clrtoeol(3XCURSES)</code>, <code>clrtoeol(3XCURSES)</code>, <code>doupdate(3XCURSES)</code>, <code>libcurses(3XCURSES)</code>, <code>refresh(3XCURSES)</code>, <code>wrefresh(3XCURSES)</code>, <code>attributes(5)</code>, <code>standards(5)</code></p> | | | | | | |

wgetbkgrnd(3XCURSES)

| | |
|----------------------|--|
| NAME | <code>bkgrnd</code> , <code>bkgrndset</code> , <code>getbkgrnd</code> , <code>wbkgrnd</code> , <code>wbkgrndset</code> , <code>wgetbkgrnd</code> – set or get the background character (and rendition) of window using a complex character |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int bkgrnd(const cchar_t *wch); void bkgrndset(const cchar_t *wch); int getbkgrnd(cchar_t *wch); int wbkgrnd(WINDOW *win, const cchar_t *wch); void wbkgrndset(WINDOW *win, const cchar_t *wch); int wgetbkgrnd(WINDOW *win, cchar_t *wch);</pre> |
| DESCRIPTION | <p>The <code>bkgrndset()</code> and <code>wbkgrndset()</code> functions turn off the previous background attributes, logical OR the requested attributes into the window rendition, and set the background property of the current or specified window based on the information in <code>wch</code>.</p> <p>The <code>bkgrnd()</code> and <code>wbkgrnd()</code> functions turn off the previous background attributes, logical OR the requested attributes into the window rendition, and set the background property of the current or specified window and then apply this setting to every character position in that window:</p> <ul style="list-style-type: none">■ The rendition of every character on the screen is changed to the new window rendition.■ Wherever the former background character appears, it is changed to the new background character. <p>If <code>wch</code> refers to a non-spacing complex character for <code>bkgrnd()</code>, <code>bkgrndset()</code>, <code>wbkgrnd()</code>, and <code>wbkgrndset()</code>, then <code>wch</code> is added to the existing spacing complex character that is the background character. If <code>wch</code> refers to a multi-column character, the results are unspecified.</p> <p>The <code>getbkgrnd()</code> and <code>wgetbkgrnd()</code> functions store, into the area pointed to by <code>wch</code>, the window's background character and rendition.</p> |
| PARAMETERS | <p><code>wch</code> Is a pointer to the complex background character to be set.</p> <p><code>win</code> Is a pointer to the window in which the complex background character is to be set.</p> |
| RETURN VALUES | <p>The <code>bkgrndset()</code> and <code>wbkgrndset()</code> functions do not return a value.</p> <p>Upon successful completion, the other functions return OK. Otherwise, they return ERR.</p> |

wgetbkgrnd(3XCURSES)

ERRORS No errors are defined.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `add_wch(3XCURSES)`, `add_wchnstr(3XCURSES)`, `addch(3XCURSES)`, `addchstr(3XCURSES)`, `attroff(3XCURSES)`, `bkgd(3XCURSES)`, `clear(3XCURSES)`, `clrtoeol(3XCURSES)`, `clrtobot(3XCURSES)`, `erase(3XCURSES)`, `inch(3XCURSES)`, `libcurses(3XCURSES)`, `mvprintw(3XCURSES)`, `attributes(5)`, `standards(5)`

wgetch(3CURSES)

| NAME | <code>curs_getch</code> , <code>getch</code> , <code>wgetch</code> , <code>mvgetch</code> , <code>mvwgetch</code> , <code>ungetch</code> – get (or push back) characters from curses terminal keyboard | | | | |
|------------------------|--|------|----------|------------------------|-----------|
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int getch(void); int wgetch(WINDOW *<i>win</i>); int mvgetch(int <i>y</i>, int <i>x</i>); int mvwgetch(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>); int ungetch(int <i>ch</i>);</pre> | | | | |
| DESCRIPTION | <p>With the <code>getch()</code>, <code>wgetch()</code>, <code>mvgetch()</code>, and <code>mvwgetch()</code> routines a character is read from the terminal associated with the window. In no-delay mode, if no input is waiting, the value <code>ERR</code> is returned. In delay mode, the program waits until the system passes text through to the program. Depending on the setting of <code>cbreak()</code>, this is after one character (<code>cbreak</code> mode), or after the first newline (<code>nocbreak</code> mode). In half-delay mode, the program waits until a character is typed or the specified timeout has been reached. Unless <code>noecho()</code> has been set, the character will also be echoed into the designated window.</p> <p>If the window is not a pad, and it has been moved or modified since the last call to <code>wrefresh()</code>, <code>wrefresh()</code> will be called before another character is read.</p> <p>If <code>keypad()</code> is <code>TRUE</code>, and a function key is pressed, the token for that function key is returned instead of the raw characters. Possible function keys are defined in <code><curses.h></code> with integers beginning with <code>0401</code>, whose names begin with <code>KEY_</code>. If a character that could be the beginning of a function key (such as escape) is received, <code>curses</code> sets a timer. If the remainder of the sequence does not come in within the designated time, the character is passed through; otherwise, the function key value is returned. For this reason, many terminals experience a delay between the time a user presses the escape key and the escape is returned to the program. Since tokens returned by these routines are outside the ASCII range, they are not printable.</p> <p>The <code>ungetch()</code> routine places <code>ch</code> back onto the input queue to be returned by the next call to <code>wgetch()</code>.</p> | | | | |
| Function Keys | <p>The following function keys, defined in <code><curses.h></code>, might be returned by <code>getch()</code> if <code>keypad()</code> has been enabled. Note that not all of these may be supported on a particular terminal if the terminal does not transmit a unique code when the key is pressed or if the definition for the key is not present in the <i>terminfo</i> database.</p> <table><thead><tr><th>Name</th><th>Key name</th></tr></thead><tbody><tr><td><code>KEY_BREAK</code></td><td>Break key</td></tr></tbody></table> | Name | Key name | <code>KEY_BREAK</code> | Break key |
| Name | Key name | | | | |
| <code>KEY_BREAK</code> | Break key | | | | |

| <i>Name</i> | <i>Key name</i> |
|-------------------|---|
| KEY_DOWN | The four arrow keys . . . |
| KEY_UP | |
| KEY_LEFT | |
| KEY_RIGHT | |
| KEY_HOME | Home key (upward+left arrow) |
| KEY_BACKSPACE | Backspace |
| KEY_F0 | Function keys; space for 64 keys is reserved. |
| KEY_F(<i>n</i>) | For $0 \leq n \leq 63$ |
| KEY_DL | Delete line |
| KEY_IL | Insert line |
| KEY_DC | Delete character |
| KEY_IC | Insert char or enter insert mode |
| KEY_EIC | Exit insert char mode |
| KEY_CLEAR | Clear screen |
| KEY_EOS | Clear to end of screen |
| KEY_EOL | Clear to end of line |
| KEY_SF | Scroll 1 line forward |
| KEY_SR | Scroll 1 line backward (reverse) |
| KEY_NPAGE | Next page |
| KEY_PPAGE | Previous page |
| KEY_STAB | Set tab |
| KEY_CTAB | Clear tab |
| KEY_CATAB | Clear all tabs |
| KEY_ENTER | Enter or send |
| KEY_SRESET | Soft (partial) reset |
| KEY_RESET | Reset or hard reset |
| KEY_PRINT | Print or copy |
| KEY_LL | Home down or bottom (lower left). Keypad is arranged like this: (Row 1) A1 up A3 (Row 2) left B2 right (Row 3) C1 down C3 |

wgetch(3CURSES)

| <i>Name</i> | <i>Key name</i> |
|---------------|-----------------------|
| KEY_A1 | Upper left of keypad |
| KEY_A3 | Upper right of keypad |
| KEY_B2 | Center of keypad |
| KEY_C1 | Lower left of keypad |
| KEY_C3 | Lower right of keypad |
| KEY_BTAB | Back tab key |
| KEY_BEG | Beg(inning) key |
| KEY_CANCEL | Cancel key |
| KEY_CLOSE | Close key |
| KEY_COMMAND | Cmd (command) key |
| KEY_COPY | Copy key |
| KEY_CREATE | Create key |
| KEY_END | End key |
| KEY_EXIT | Exit key |
| KEY_FIND | Find key |
| KEY_HELP | Help key |
| KEY_MARK | Mark key |
| KEY_MESSAGE | Message key |
| KEY_MOVE | Move key |
| KEY_NEXT | Next object key |
| KEY_OPEN | Open key |
| KEY_OPTIONS | Options key |
| KEY_PREVIOUS | Previous object key |
| KEY_REDO | Redo key |
| KEY_REFERENCE | Reference key |
| KEY_REFRESH | Refresh key |
| KEY_REPLACE | Replace key |
| KEY_RESTART | Restart key |
| KEY_RESUME | Resume key |

| <i>Name</i> | <i>Key name</i> |
|---------------|-------------------------|
| KEY_SAVE | Save key |
| KEY_SBEG | Shifted beginning key |
| KEY_SCANCEL | Shifted cancel key |
| KEY_SCOMMAND | Shifted command key |
| KEY_SCOPY | Shifted copy key |
| KEY_SCREATE | Shifted create key |
| KEY_SDC | Shifted delete char key |
| KEY_SDL | Shifted delete line key |
| KEY_SELECT | Select key |
| KEY_SEND | Shifted end key |
| KEY_SEOL | Shifted clear line key |
| KEY_SEXIT | Shifted exit key |
| KEY_SFIND | Shifted find key |
| KEY_SHELP | Shifted help key |
| KEY_SHOME | Shifted home key |
| KEY_SIC | Shifted input key |
| KEY_SLEFT | Shifted left arrow key |
| KEY_SMESSAGE | Shifted message key |
| KEY_SMOVE | Shifted move key |
| KEY_SNEXT | Shifted next key |
| KEY_SOPTIONS | Shifted options key |
| KEY_SPREVIOUS | Shifted prev key |
| KEY_SPRINT | Shifted print key |
| KEY_SREDO | Shifted redo key |
| KEY_SREPLACE | Shifted replace key |
| KEY_SRIGHT | Shifted right arrow |
| KEY_SRSUME | Shifted resume key |
| KEY_SSAVE | Shifted save key |
| KEY_SSUSPEND | Shifted suspend key |

wgetch(3CURSES)

| <i>Name</i> | <i>Key name</i> |
|-------------|------------------|
| KEY_SUNDO | Shifted undo key |
| KEY_SUSPEND | Suspend key |
| KEY_UNDO | Undo key |

RETURN VALUES All routines return the integer ERR upon failure. The wgetch() routine returns an integer value other than ERR upon successful completion. The other routines return the next input character or function key code upon successful completion.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO curs_inopts(3CURSES), curs_move(3CURSES), curs_refresh(3CURSES), curses(3CURSES), attributes(5)

NOTES The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.

Use of the escape key for a single character function is discouraged.

When using getch(), wgetch(), mvgetch(), or mvwgetch(), nocbreak mode (nocbreak()) and echo mode (echo()) should not be used at the same time. Depending on the state of the tty driver when each character is typed, the program may produce undesirable results.

Note that getch(), mvgetch(), and mvwgetch() may be macros.

| | |
|--|---|
| NAME | getch, wgetch, mvgetch, mvwgetch – get a single-byte character from the terminal |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int getch(void); int wgetch(WINDOW *win); int mvgetch(int y, int x); int mvwgetch(WINDOW *win, int y, int x);</pre> |
| PARAMETERS | <p><i>win</i> Is a pointer to the window associated with the terminal from which the character is to be read.</p> <p><i>y</i> Is the y (row) coordinate for the position of the character to be read.</p> <p><i>x</i> Is the x (column) coordinate for the position of the character to be read.</p> |
| DESCRIPTION | <p>These functions read a single-byte character from the terminal associated with the current or specified window. The results are unspecified if the input is not a single-byte character. If keypad(3XCURSES) is enabled, these functions respond to the pressing of a function key by returning the corresponding KEY_ value defined in <curses.h></p> <p>Processing of terminal input is subject to the general rules described on the keypad(3XCURSES) manual page.</p> <p>If echoing is enabled, then the character is echoed as though it were provided as an input argument to addch(3XCURSES), except for the following characters:</p> <p><backspace> The input is interpreted as follows: unless the cursor already was in column 0, <backspace> moves the cursor one column toward the start of the current line and any characters after the <backspace> are added or inserted starting there. The character at the resulting cursor position is then deleted as though delch(3XCURSES) were called, except that if the cursor was originally in the first column of the line, the user is alerted as though beep(3XCURSES) were called.</p> <p>Function keys The user is alerted as though beep() were called. Information concerning the function keys is not returned to the caller.</p> <p>If the current or specified window is not a pad, and it has been moved modified since the last refresh operation, then it will be refreshed before another character is read.</p> <p>The following is a list of tokens for function keys that are returned by the getch() set of functions if keypad handling is enabled (some terminals may not support all tokens).</p> |
| Constant Values for Function Keys | |

wgetch(3XCURSES)

| Constant | Description |
|-------------------|---|
| KEY_BREAK | Break key |
| KEY_DOWN | The down arrow key |
| KEY_UP | The up arrow key |
| KEY_LEFT | The left arrow key |
| KEY_RIGHT | The right arrow key |
| KEY_HOME | Home key |
| KEY_BACKSPACE | Backspace |
| KEY_F0 | Function keys. Space for 64 keys is reserved. |
| KEY_F(<i>n</i>) | For $0 \leq n \leq 63$ |
| KEY_DL | Delete line |
| KEY_IL | Insert line |
| KEY_DC | Delete character |
| KEY_IC | Insert char or enter insert mode |
| KEY_EIC | Exit insert char mode |
| KEY_CLEAR | Clear screen |
| KEY_EOS | Clear to end of screen |
| KEY_EOL | Clear to end of line |
| KEY_SF | Scroll 1 line forward |
| KEY_SR | Scroll 1 line backwards |
| KEY_NPAGE | Next page |
| KEY_PPAGE | Previous page |
| KEY_STAB | Set tab |
| KEY_CTAB | Clear tab |
| KEY_CATAB | Clear all tabs |
| KEY_ENTER | Enter or send |
| KEY_SRESET | Soft (partial) reset |
| KEY_RESET | Reset or hard reset |
| KEY_PRINT | Print or copy |
| KEY_LL | Home down or bottom (lower left) |

| Constant | Description |
|---------------|-----------------------|
| KEY_A1 | Upper left of keypad |
| KEY_A3 | Upper right of keypad |
| KEY_B2 | Center of keypad |
| KEY_C1 | Lower left of keypad |
| KEY_C3 | Lower right of keypad |
| KEY_BTAB | Back tab |
| KEY_BEG | Beginning key |
| KEY_CANCEL | Cancel key |
| KEY_CLOSE | Close key |
| KEY_COMMAND | Cmd (command) key |
| KEY_COPY | Copy key |
| KEY_CREATE | Create key |
| KEY_END | End key |
| KEY_EXIT | Exit key |
| KEY_FIND | Find key |
| KEY_HELP | Help key |
| KEY_MARK | Mark key |
| KEY_MESSAGE | Message key |
| KEY_MOVE | Move key |
| KEY_NEXT | Next object key |
| KEY_OPEN | Open key |
| KEY_OPTIONS | Options key |
| KEY_PREVIOUS | Previous object key |
| KEY_REDO | Redo key |
| KEY_REFERENCE | Reference key |
| KEY_REFRESH | Refresh key |
| KEY_REPLACE | Replace key |
| KEY_RESTART | Restart key |
| KEY_RESUME | Resume key |

wgetch(3XCURSES)

| Constant | Description |
|---------------|-------------------------|
| KEY_SAVE | Save key |
| KEY_SBEG | Shifted beginning key |
| KEY_SCANCEL | Shifted cancel key |
| KEY_SCOMMAND | Shifted command key |
| KEY_SCOPY | Shifted copy key |
| KEY_SCREATE | Shifted create key |
| KEY_SDC | Shifted delete char key |
| KEY_SDL | Shifted delete line key |
| KEY_SELECT | Select key |
| KEY_SEND | Shifted end key |
| KEY_SEOL | Shifted clear line key |
| KEY_SEXIT | Shifted exit key |
| KEY_SFIND | Shifted find key |
| KEY_SHELP | Shifted help key |
| KEY_SHOME | Shifted home key |
| KEY_SIC | Shifted input key |
| KEY_SLEFT | Shifted left arrow key |
| KEY_SMESSAGES | Shifted messages key |
| KEY_SMOVE | Shifted move key |
| KEY_SNEXT | Shifted next key |
| KEY_SOPTIONS | Shifted options key |
| KEY_SPREVIOUS | Shifted previous key |
| KEY_SPRINT | Shifted print key |
| KEY_SREDO | Shifted redo key |
| KEY_SREPLACE | Shifted replace key |
| KEY_SRIGHT | Shifted right arrow key |
| KEY_SRSUME | Shifted resume key |
| KEY_SSAVE | Shifted save key |
| KEY_SSUSPEND | Shifted suspend key |

| Constant | Description |
|-------------|------------------|
| KEY_SUNDO | Shifted undo key |
| KEY_SUSPEND | Suspend key |
| KEY_UNDO | Undo key |

RETURN VALUES Upon successful completion, these functions return the single-byte character, `KEY_` value, or `ERR`. When in the nodelay mode and no data is available, `ERR` is returned.

ERRORS No errors are defined.

USAGE Applications should not define the escape key by itself as a single-character function.

When using these functions, nocbreak mode (`cbreak(3XCURSES)`) and echo mode (`echo(3XCURSES)`) should not be used at the same time. Depending on the state of the terminal when each character is typed, the application may produce undesirable results.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `cbreak(3XCURSES)`, `echo(3XCURSES)`, `halfdelay(3XCURSES)`, `keypad(3XCURSES)`, `libcurses(3XCURSES)`, `nodelay(3XCURSES)`, `notimeout(3XCURSES)`, `raw(3XCURSES)`, `timeout(3XCURSES)`, `attributes(5)`, `standards(5)`

wgetnstr(3CURSES)

| NAME | <code> curs_getstr, getstr, wgetstr, mvgetstr, mvwgetstr, wgetnstr</code> – get character strings from curses terminal keyboard | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int getstr(char *str); int wgetstr(WINDOW *win, char *str); int mvgetstr(int y, int x, char *str); int mvwgetstr(WINDOW *win, int y, int x, char *str); int wgetnstr(WINDOW *win, char *str, int n);</pre> | | | | |
| DESCRIPTION | The effect of <code>getstr()</code> is as though a series of calls to <code>getch()</code> were made, until a newline or carriage return is received. The resulting value is placed in the area pointed to by the character pointer <code>str</code> . <code>wgetnstr()</code> reads at most <code>n</code> characters, thus preventing a possible overflow of the input buffer. The user's erase and kill characters are interpreted, as well as any special keys (such as function keys, HOME key, and CLEAR key.) | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curs_getch(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . Note that <code>getstr()</code> , <code>mvgetstr()</code> , and <code>mvwgetstr()</code> may be macros. | | | | |

| | | | | | | | | | | | |
|--------------------|--|------------|--|----------|---|----------|--|----------|---|------------|---|
| NAME | getnstr, getstr, mvgetnstr, mvgetstr, mvwgetnstr, mvwgetstr, wgetnstr, wgetstr – get a multibyte character string from terminal | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int getnstr(char *str, int n); int getstr(char *str); int mvgetnstr(int y, int x, char *str, int n); int mvgetstr(int y, int x, char *str); int mvwgetnstr(WINDOW *win, int y, int x, char *str, int n); int mvwgetstr(WINDOW *win, int y, int x, char *str); int wgetnstr(WINDOW *win, char *str, int n); int wgetstr(WINDOW *win, char *str);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The <code>getstr()</code> and <code>wgetstr()</code> functions get a character string from the terminal associated with the window <code>stdscr</code> or window <code>win</code>, respectively. The <code>mvgetstr()</code> and <code>mvwgetstr()</code> functions move the cursor to the position specified in <code>stdscr</code> or <code>win</code>, respectively, then get a character string.</p> <p>These functions call <code>wgetch(3XCURSES)</code> and place each received character in <code>str</code> until a newline is received, which is also placed in <code>str</code>. The erase and kill characters set by the user are processed.</p> <p>The <code>getnstr()</code>, <code>mvgetnstr()</code>, <code>mvwgetnstr()</code> and <code>wgetnstr()</code> functions read at most <code>n</code> characters. These functions are used to prevent overflowing the input buffer.</p> <p>The <code>getnstr()</code>, <code>wgetnstr()</code>, <code>mvgetnstr()</code>, and <code>mvwgetnstr()</code> functions only return complete multibyte characters. If the area pointed to by <code>str</code> is not large enough to hold at least one character, these functions fail.</p> | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>str</i></td> <td>Is a pointer to the area where the character string is to be placed.</td> </tr> <tr> <td><i>n</i></td> <td>Is the maximum number of characters to read from input.</td> </tr> <tr> <td><i>y</i></td> <td>Is the y (row) coordinate of starting position of character string to be read.</td> </tr> <tr> <td><i>x</i></td> <td>Is the x (column) coordinate of starting position of character string to be read.</td> </tr> <tr> <td><i>win</i></td> <td>Points to the window associated with the terminal from which the character is to be read.</td> </tr> </table> | <i>str</i> | Is a pointer to the area where the character string is to be placed. | <i>n</i> | Is the maximum number of characters to read from input. | <i>y</i> | Is the y (row) coordinate of starting position of character string to be read. | <i>x</i> | Is the x (column) coordinate of starting position of character string to be read. | <i>win</i> | Points to the window associated with the terminal from which the character is to be read. |
| <i>str</i> | Is a pointer to the area where the character string is to be placed. | | | | | | | | | | |
| <i>n</i> | Is the maximum number of characters to read from input. | | | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of starting position of character string to be read. | | | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of starting position of character string to be read. | | | | | | | | | | |
| <i>win</i> | Points to the window associated with the terminal from which the character is to be read. | | | | | | | | | | |

wgetnstr(3XCURSES)

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `getch(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_getwstr, getwstr, getnwstr, wgetwstr, wgetnwstr, mvgetwstr, mvgetnwstr, mvwgetwstr, mvwgetnwstr – get wchar_t character strings from curses terminal keyboard | | | | |
|---------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int getwstr(wchar_t *wstr); int getnwstr(wchar_t *wstr, int n); int wgetwstr(WINDOW *win, wchar_t *wstr); int wgetnwstr(WINDOW *win, wchar_t *wstr, int n); int mvgetwstr(int y, int x, wchar_t *wstr); int mvgetnwstr(int y, int x, wchar_t *wstr, int n); int mvwgetwstr(WINDOW *win, int y, int x, wchar_t *wstr); int mvwgetnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);</pre> | | | | |
| DESCRIPTION | The effect of getwstr() is as though a series of calls to getwch(3CURSES) were made, until a newline and carriage return is received. The resulting value is placed in the area pointed to by the wchar_t pointer wstr. getnwstr() reads at most n wchar_t characters, thus preventing a possible overflow of the input buffer. The user's erase and kill characters are interpreted, as well as any special keys (such as function keys, HOME key, CLEAR key, etc.). | | | | |
| RETURN VALUE | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See attributes(5) for a description of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), getwch(3CURSES), attributes(5) | | | | |
| NOTES | <p>The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h>, and <wctype.h>.</p> <p>Note that all routines except wgetnwstr() may be macros.</p> | | | | |

wgetn_wstr(3XCURSES)

| | |
|----------------------|---|
| NAME | getn_wstr, get_wstr, mvgetn_wstr, mvget_wstr, mvwgetn_wstr, mvwget_wstr, wgetn_wstr, wget_wstr – get a wide character string from terminal |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int getn_wstr(wint_t *wstr, int n); int get_wstr(wint_t *wstr); int mvgetn_wstr(int y, int x, wint_t *wstr, int n); int mvget_wstr(int y, int x, wint_t *wstr); int mvwgetn_wstr(WINDOW *win, int y, int x, wint_t *wstr, int n); int mvwget_wstr(WINDOW *win, int y, int x, wint_t *wstr); int wgetn_wstr(WINDOW *win, wint_t *wstr, int n); int wget_wstr(WINDOW *win, wint_t *wstr);</pre> |
| DESCRIPTION | <p>The <code>get_wstr()</code> and <code>wget_wstr()</code> functions get a wide character string from the terminal associated with the window <code>stdscr</code> or window <code>win</code>, respectively. The <code>mvget_wstr()</code> and <code>mvwget_wstr()</code> functions move the cursor to the position specified in <code>stdscr</code> or <code>win</code>, respectively, then get a wide character string.</p> <p>These functions call <code>wget_wch(3XCURSES)</code> and place each received character in <code>wstr</code> until a newline character, end-of-line character, or end-of-file character is received, which is also placed in <code>wstr</code>. The erase and kill characters set by the user are processed.</p> <p>The <code>getn_wstr()</code>, <code>mvgetn_wstr()</code>, <code>mvwgetn_wstr()</code> and <code>wgetn_wstr()</code> functions read at most <code>n</code> characters. These functions are used to prevent overflowing the input buffer.</p> |
| PARAMETERS | <p><i>wstr</i> Is a pointer to the area where the character string is to be placed.</p> <p><i>n</i> Is the maximum number of characters to read from input.</p> <p><i>y</i> Is the y (row) coordinate of starting position of character string to be read.</p> <p><i>x</i> Is the x (column) coordinate of starting position of character string to be read.</p> <p><i>win</i> points to the window associated with the terminal from which the character is to be read.</p> |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. |
| ERRORS | None. |

wgetn_wstr(3XCURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `get_wch(3XCURSES)`, `getnstr(3XCURSES)`, `libcurses(3XCURSES)`,
`attributes(5)`, `standards(5)`

wgetstr(3CURSES)

| NAME | <code> curs_getstr, getstr, wgetstr, mvgetstr, mvwgetstr, wgetnstr </code> – get character strings from curses terminal keyboard | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [<i>flag</i> ...] <i>file</i> ... -lcurses [<i>library</i> ...] #include <curses.h> int getstr(char *<i>str</i>); int wgetstr(WINDOW *<i>win</i>, char *<i>str</i>); int mvgetstr(int <i>y</i>, int <i>x</i>, char *<i>str</i>); int mvwgetstr(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>, char *<i>str</i>); int wgetnstr(WINDOW *<i>win</i>, char *<i>str</i>, int <i>n</i>);</pre> | | | | |
| DESCRIPTION | The effect of <code>getstr()</code> is as though a series of calls to <code>getch()</code> were made, until a newline or carriage return is received. The resulting value is placed in the area pointed to by the character pointer <i>str</i> . <code>wgetnstr()</code> reads at most <i>n</i> characters, thus preventing a possible overflow of the input buffer. The user's erase and kill characters are interpreted, as well as any special keys (such as function keys, HOME key, and CLEAR key.) | | | | |
| RETURN VALUES | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code> curs_getch(3CURSES), curses(3CURSES), attributes(5) </code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . Note that <code>getstr()</code> , <code>mvgetstr()</code> , and <code>mvwgetstr()</code> may be macros. | | | | |

| | | | | | | | | | | | |
|--------------------|--|------------|--|----------|---|----------|--|----------|---|------------|---|
| NAME | getnstr, getstr, mvgetnstr, mvgetstr, mvwgetnstr, mvwgetstr, wgetnstr, wgetstr – get a multibyte character string from terminal | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int getnstr(char *str, int n); int getstr(char *str); int mvgetnstr(int y, int x, char *str, int n); int mvgetstr(int y, int x, char *str); int mvwgetnstr(WINDOW *win, int y, int x, char *str, int n); int mvwgetstr(WINDOW *win, int y, int x, char *str); int wgetnstr(WINDOW *win, char *str, int n); int wgetstr(WINDOW *win, char *str);</pre> | | | | | | | | | | |
| DESCRIPTION | <p>The <code>getstr()</code> and <code>wgetstr()</code> functions get a character string from the terminal associated with the window <code>stdscr</code> or window <code>win</code>, respectively. The <code>mvgetstr()</code> and <code>mvwgetstr()</code> functions move the cursor to the position specified in <code>stdscr</code> or <code>win</code>, respectively, then get a character string.</p> <p>These functions call <code>wgetch(3XCURSES)</code> and place each received character in <code>str</code> until a newline is received, which is also placed in <code>str</code>. The erase and kill characters set by the user are processed.</p> <p>The <code>getnstr()</code>, <code>mvgetnstr()</code>, <code>mvwgetnstr()</code> and <code>wgetnstr()</code> functions read at most <code>n</code> characters. These functions are used to prevent overflowing the input buffer.</p> <p>The <code>getnstr()</code>, <code>wgetnstr()</code>, <code>mvgetnstr()</code>, and <code>mvwgetnstr()</code> functions only return complete multibyte characters. If the area pointed to by <code>str</code> is not large enough to hold at least one character, these functions fail.</p> | | | | | | | | | | |
| PARAMETERS | <table border="0"> <tr> <td style="padding-right: 20px;"><i>str</i></td> <td>Is a pointer to the area where the character string is to be placed.</td> </tr> <tr> <td><i>n</i></td> <td>Is the maximum number of characters to read from input.</td> </tr> <tr> <td><i>y</i></td> <td>Is the y (row) coordinate of starting position of character string to be read.</td> </tr> <tr> <td><i>x</i></td> <td>Is the x (column) coordinate of starting position of character string to be read.</td> </tr> <tr> <td><i>win</i></td> <td>Points to the window associated with the terminal from which the character is to be read.</td> </tr> </table> | <i>str</i> | Is a pointer to the area where the character string is to be placed. | <i>n</i> | Is the maximum number of characters to read from input. | <i>y</i> | Is the y (row) coordinate of starting position of character string to be read. | <i>x</i> | Is the x (column) coordinate of starting position of character string to be read. | <i>win</i> | Points to the window associated with the terminal from which the character is to be read. |
| <i>str</i> | Is a pointer to the area where the character string is to be placed. | | | | | | | | | | |
| <i>n</i> | Is the maximum number of characters to read from input. | | | | | | | | | | |
| <i>y</i> | Is the y (row) coordinate of starting position of character string to be read. | | | | | | | | | | |
| <i>x</i> | Is the x (column) coordinate of starting position of character string to be read. | | | | | | | | | | |
| <i>win</i> | Points to the window associated with the terminal from which the character is to be read. | | | | | | | | | | |

wgetstr(3XCURSES)

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `getch(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_getwch, getwch, wgetwch, mvgetwch, mvwgetwch, ungetwch – get (or push back) wchar_t characters from curses terminal keyboard | | | | |
|----------------------|--|------|----------|-----------|-----------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int getwch(void); int wgetwch(WINDOW *win); int mvgetwch(int y, int x); int mvwgetwch(WINDOW *win, int y, int x); int ungetwch(int wch);</pre> | | | | |
| DESCRIPTION | <p>The <code>getwch()</code>, <code>wgetwch()</code>, <code>mvgetwch()</code>, and <code>mvwgetwch()</code> routines read an EUC character from the terminal associated with the window, transform it into a <code>wchar_t</code> character, and return a <code>wchar_t</code> character. In no-delay mode, if no input is waiting, the value <code>ERR</code> is returned. In delay mode, the program waits until the system passes text through to the program. Depending on the setting of <code>cbreak</code>, this is after one character (<code>cbreak</code> mode), or after the first newline (<code>nocbreak</code> mode). In half-delay mode, the program waits until a character is typed or the specified timeout has been reached. Unless <code>noecho</code> has been set, the character will also be echoed into the designated window.</p> <p>If the window is not a pad, and it has been moved or modified since the last call to <code>wrefresh(3CURSES)</code>, <code>wrefresh</code> will be called before another character is read.</p> <p>If <code>keypad</code> is <code>TRUE</code>, and a function key is pressed, the token for that function key is returned instead of the raw characters. Possible function keys are defined in <code><curses.h></code> with integers beginning with <code>0401</code>, whose names begin with <code>KEY_</code>. If a character that could be the beginning of a function key (such as escape) is received, <code>curses(3CURSES)</code> sets a timer. If the remainder of the sequence does not come in within the designated time, the character is passed through; otherwise, the function key value is returned. For this reason, many terminals experience a delay between the time a user presses the escape key and the escape is returned to the program.</p> <p>The <code>ungetwch()</code> routine places <code>wch</code> back onto the input queue to be returned by the next call to <code>wgetwch()</code>.</p> | | | | |
| Function Keys | <p>The following function keys, defined in <code><curses.h></code>, might be returned by <code>getwch()</code> if <code>keypad</code> has been enabled. Note that not all of these may be supported on a particular terminal if the terminal does not transmit a unique code when the key is pressed or if the definition for the key is not present in the <code>terminfo(4)</code> database.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Name</th> <th style="text-align: left; padding: 2px;">Key name</th> </tr> </thead> <tbody> <tr> <td style="padding: 2px;">KEY_BREAK</td> <td style="padding: 2px;">Break key</td> </tr> </tbody> </table> | Name | Key name | KEY_BREAK | Break key |
| Name | Key name | | | | |
| KEY_BREAK | Break key | | | | |

wgetwch(3CURSES)

| <i>Name</i> | <i>Key name</i> |
|-------------------|---|
| KEY_DOWN | The four arrow keys . . . |
| KEY_UP | |
| KEY_LEFT | |
| KEY_RIGHT | |
| KEY_HOME | Home key (upward+left arrow) |
| KEY_BACKSPACE | Backspace |
| KEY_F0 | Function keys; space for 64 keys is reserved. |
| KEY_F(<i>n</i>) | For $0 \leq n \leq 63$ |
| KEY_DL | Delete line |
| KEY_IL | Insert line |
| KEY_DC | Delete character |
| KEY_IC | Insert char or enter insert mode |
| KEY_EIC | Exit insert char mode |
| KEY_CLEAR | Clear screen |
| KEY_EOS | Clear to end of screen |
| KEY_EOL | Clear to end of line |
| KEY_SF | Scroll 1 line forward |
| KEY_SR | Scroll 1 line backward (reverse) |
| KEY_NPAGE | Next page |
| KEY_PPAGE | Previous page |
| KEY_STAB | Set tab |
| KEY_CTAB | Clear tab |
| KEY_CATAB | Clear all tabs |
| KEY_ENTER | Enter or send |
| KEY_SRESET | Soft (partial) reset |
| KEY_RESET | Reset or hard reset |
| KEY_PRINT | Print or copy |
| KEY_LL | Home down or bottom (lower left). Keypad is arranged like this: A1 up A3 left B2 right C1 down C3 |

| <i>Name</i> | <i>Key name</i> |
|---------------|-----------------------|
| KEY_A1 | Upper left of keypad |
| KEY_A3 | Upper right of keypad |
| KEY_B2 | Center of keypad |
| KEY_C1 | Lower left of keypad |
| KEY_C3 | Lower right of keypad |
| KEY_BTAB | Back tab key |
| KEY_BEG | Beg(inning) key |
| KEY_CANCEL | Cancel key |
| KEY_CLOSE | Close key |
| KEY_COMMAND | Cmd (command) key |
| KEY_COPY | Copy key |
| KEY_CREATE | Create key |
| KEY_END | End key |
| KEY_EXIT | Exit key |
| KEY_FIND | Find key |
| KEY_HELP | Help key |
| KEY_MARK | Mark key |
| KEY_MESSAGE | Message key |
| KEY_MOVE | Move key |
| KEY_NEXT | Next object key |
| KEY_OPEN | Open key |
| KEY_OPTIONS | Options key |
| KEY_PREVIOUS | Previous object key |
| KEY_REDO | Redo key |
| KEY_REFERENCE | Reference key |
| KEY_REFRESH | Refresh key |
| KEY_REPLACE | Replace key |
| KEY_RESTART | Restart key |
| KEY_RESUME | Resume key |

wgetwch(3CURSES)

| <i>Name</i> | <i>Key name</i> |
|---------------|-------------------------|
| KEY_SAVE | Save key |
| KEY_SBEG | Shifted beginning key |
| KEY_SCANCEL | Shifted cancel key |
| KEY_SCOMMAND | Shifted command key |
| KEY_SCOPY | Shifted copy key |
| KEY_SCREATE | Shifted create key |
| KEY_SDC | Shifted delete char key |
| KEY_SDL | Shifted delete line key |
| KEY_SELECT | Select key |
| KEY_SEND | Shifted end key |
| KEY_SEOL | Shifted clear line key |
| KEY_SEXIT | Shifted exit key |
| KEY_SFIND | Shifted find key |
| KEY_SHELP | Shifted help key |
| KEY_SHOME | Shifted home key |
| KEY_SIC | Shifted input key |
| KEY_SLEFT | Shifted left arrow key |
| KEY_SMESSAGE | Shifted message key |
| KEY_SMOVE | Shifted move key |
| KEY_SNEXT | Shifted next key |
| KEY_SOPTIONS | Shifted options key |
| KEY_SPREVIOUS | Shifted prev key |
| KEY_SPRINT | Shifted print key |
| KEY_SREDO | Shifted redo key |
| KEY_SREPLACE | Shifted replace key |
| KEY_SRIGHT | Shifted right arrow |
| KEY_SRSUME | Shifted resume key |
| KEY_SSAVE | Shifted save key |
| KEY_SSUSPEND | Shifted suspend key |

| <i>Name</i> | <i>Key name</i> |
|-------------|------------------|
| KEY_SUNDO | Shifted undo key |
| KEY_SUSPEND | Suspend key |
| KEY_UNDO | Undo key |

RETURN VALUE All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curses(3CURSES)`, `curl_inopts(3CURSES)`, `curl_move(3CURSES)`, `wrefresh(3CURSES)`, `terminfo(4)`, `attributes(5)`

NOTES The header file `<curses.h>` automatically includes the header files `<stdio.h>`, `<unctrl.h>` and `<widec.h>`.

Use of the escape key by a programmer for a single character function is discouraged.

When using `getwch()`, `wgetwch()`, `mvgetwch()`, or `mvwgetwch()`, `nocbreak` mode and `echo` mode should not be used at the same time. Depending on the state of the tty driver when each character is typed, the program may produce undesirable results.

Note that `getwch()`, `mvgetwch()`, and `mvwgetwch()` may be macros.

wget_wch(3XCURSES)

| | |
|--------------------|---|
| NAME | get_wch, wget_wch, mvget_wch, mvwget_wch – get a wide character from terminal |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int get_wch(wint_t *ch); int wget_wch(WINDOW *win, wint_t *ch); int mvget_wch(int y, int x, wint_t *ch); int mvwget_wch(WINDOW *win, int y, int x, wint_t *ch);</pre> |
| DESCRIPTION | <p>The <code>get_wch()</code> and <code>wget_wch()</code> functions get a wide character from the terminal associated with the window <code>stdscr</code> or window <code>win</code>, respectively. The <code>mvget_wch()</code> and <code>mvwget_wch()</code> functions move the cursor to the position specified in <code>stdscr</code> or <code>win</code>, respectively, then get a character.</p> <p>If the window is not a pad and has been changed since the last call to <code>refresh(3XCURSES)</code>, <code>get_wch()</code> calls <code>refresh()</code> to update the window before the next character is read.</p> <p>The setting of certain functions affects the behavior of the <code>get_wch()</code> set of functions. For example, if <code>cbreak(3XCURSES)</code> is set, characters typed by the user are immediately processed. If <code>halfdelay(3XCURSES)</code> is set, <code>get_wch()</code> waits until a character is typed or returns <code>ERR</code> if no character is typed within the specified timeout period. This timeout can also be specified for individual windows with the <i>delay</i> parameter of <code>timeout(3XCURSES)</code>. A negative value waits for input; a value of 0 returns <code>ERR</code> if no input is ready; a positive value blocks until input arrives or the time specified expires (in which case <code>ERR</code> is returned). If <code>nodelay(3XCURSES)</code> is set, <code>ERR</code> is returned if no input is waiting; if not set, <code>get_wch()</code> waits until input arrives. Each character will be echoed to the window unless <code>noecho(3XCURSES)</code> has been set.</p> <p>If keypad handling is enabled (<code>keypad(3XCURSES)</code> is <code>TRUE</code>), the token for the function key (a <code>KEY_</code> value) is stored in the object pointed to by <code>ch</code> and <code>KEY_CODE_YES</code> is returned. If a character is received that could be the beginning of a function key (for example, <code>ESC</code>), an inter-byte timer is set. If the remainder of the sequence is not received before the time expires, the character is passed through; otherwise, the value of the function key is returned. If <code>notimeout()</code> is set, the inter-byte timer is not used.</p> <p>The <code>ESC</code> key is typically a prefix key used with function keys and should not be used as a single character.</p> <p>See the <code>getch(3XCURSES)</code> manual page for a list of tokens for function keys that are returned by the <code>get_wch()</code> set of functions if keypad handling is enabled (Some terminals may not support all tokens).</p> |

wget_wch(3XCURSES)

PARAMETERS

ch Is a pointer to a wide integer where the returned wide character or KEY_ value can be stored.

win Is a pointer to the window associated with the terminal from which the character is to be read.

y Is the y (row) coordinate for the position of the character to be read.

x Is the x (column) coordinate for the position of the character to be read.

RETURN VALUES When these functions successfully report the pressing of a function key, they return KEY_CODE_YES. When they successfully report a wide character, they return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO cbreak(3XCURSES), echo(3XCURSES), halfdelay(3XCURSES), keypad(3XCURSES), libcurses(3XCURSES), nodelay(3XCURSES), notimeout(3XCURSES), raw(3XCURSES), timeout(3XCURSES), attributes(5), standards(5)

wgetwstr(3CURSES)

| NAME | <code>curs_getwstr</code> , <code>getwstr</code> , <code>getnwstr</code> , <code>wgetwstr</code> , <code>wgetnwstr</code> , <code>mvgetwstr</code> , <code>mvgetnwstr</code> , <code>mvwgetwstr</code> , <code>mvwgetnwstr</code> – get <code>wchar_t</code> character strings from curses terminal keyboard | | | | |
|---------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ..] #include <curses.h> int getwstr(wchar_t *wstr); int getnwstr(wchar_t *wstr, int n); int wgetwstr(WINDOW *win, wchar_t *wstr); int wgetnwstr(WINDOW *win, wchar_t *wstr, int n); int mvgetwstr(int y, int x, wchar_t *wstr); int mvgetnwstr(int y, int x, wchar_t *wstr, int n); int mvwgetwstr(WINDOW *win, int y, int x, wchar_t *wstr); int mvwgetnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);</pre> | | | | |
| DESCRIPTION | The effect of <code>getwstr()</code> is as though a series of calls to <code>getwch(3CURSES)</code> were made, until a newline and carriage return is received. The resulting value is placed in the area pointed to by the <code>wchar_t</code> pointer <code>wstr</code> . <code>getnwstr()</code> reads at most <code>n</code> <code>wchar_t</code> characters, thus preventing a possible overflow of the input buffer. The user's erase and kill characters are interpreted, as well as any special keys (such as function keys, HOME key, CLEAR key, etc.). | | | | |
| RETURN VALUE | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for a description of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>getwch(3CURSES)</code> , <code>attributes(5)</code> | | | | |
| NOTES | The header file <code><curses.h></code> automatically includes the header files <code><stdio.h></code> , <code><unctrl.h></code> , and <code><widec.h></code> . Note that all routines except <code>wgetnwstr()</code> may be macros. | | | | |

| | |
|----------------------|---|
| NAME | getn_wstr, get_wstr, mvgetn_wstr, mvget_wstr, mvwgetn_wstr, mvwget_wstr, wgetn_wstr, wget_wstr – get a wide character string from terminal |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int getn_wstr(wint_t *wstr, int n); int get_wstr(wint_t *wstr); int mvgetn_wstr(int y, int x, wint_t *wstr, int n); int mvget_wstr(int y, int x, wint_t *wstr); int mvwgetn_wstr(WINDOW *win, int y, int x, wint_t *wstr, int n); int mvwget_wstr(WINDOW *win, int y, int x, wint_t *wstr); int wgetn_wstr(WINDOW *win, wint_t *wstr, int n); int wget_wstr(WINDOW *win, wint_t *wstr);</pre> |
| DESCRIPTION | <p>The <code>get_wstr()</code> and <code>wget_wstr()</code> functions get a wide character string from the terminal associated with the window <code>stdscr</code> or window <code>win</code>, respectively. The <code>mvget_wstr()</code> and <code>mvwget_wstr()</code> functions move the cursor to the position specified in <code>stdscr</code> or <code>win</code>, respectively, then get a wide character string.</p> <p>These functions call <code>wget_wch(3XCURSES)</code> and place each received character in <code>wstr</code> until a newline character, end-of-line character, or end-of-file character is received, which is also placed in <code>wstr</code>. The erase and kill characters set by the user are processed.</p> <p>The <code>getn_wstr()</code>, <code>mvgetn_wstr()</code>, <code>mvwgetn_wstr()</code> and <code>wgetn_wstr()</code> functions read at most <code>n</code> characters. These functions are used to prevent overflowing the input buffer.</p> |
| PARAMETERS | <p><i>wstr</i> Is a pointer to the area where the character string is to be placed.</p> <p><i>n</i> Is the maximum number of characters to read from input.</p> <p><i>y</i> Is the y (row) coordinate of starting position of character string to be read.</p> <p><i>x</i> Is the x (column) coordinate of starting position of character string to be read.</p> <p><i>win</i> points to the window associated with the terminal from which the character is to be read.</p> |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. |
| ERRORS | None. |

wget_wstr(3XCURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `get_wch(3XCURSES)`, `getnstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| | | | | | | | | | | | | | | | | | |
|--------------------|---|-----------|-------------------------|-----------|--------------------------|-----------|------------------------|-----------|---------------------------|-----------|----------------------|-----------|-----------------------|-----------|-------------------------|-----------|--------------------------|
| NAME | curs_border, border, wborder, box, whline, wvline – create curses borders, horizontal and vertical lines | | | | | | | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int border(chtype <i>ls</i>, chtype <i>rs</i>, chtype <i>ts</i>, chtype <i>bs</i>, chtype <i>tl</i>, chtype <i>tr</i>, chtype <i>bl</i>, chtype <i>br</i>); int wborder(WINDOW *<i>win</i>, chtype <i>ls</i>, chtype <i>rs</i>, chtype <i>ts</i>, chtype <i>bs</i>, chtype <i>tl</i>, chtype <i>tr</i>, chtype <i>bl</i>, chtype <i>br</i>); int box(WINDOW *<i>win</i>, chtype <i>verch</i>, chtype <i>horch</i>); int hline(chtype <i>ch</i>, int <i>n</i>); int whline(WINDOW *<i>win</i>, chtype <i>ch</i>, int <i>n</i>); int vline(chtype <i>ch</i>, int <i>n</i>); int wvline(WINDOW *<i>win</i>, chtype <i>ch</i>, int <i>n</i>);</pre> | | | | | | | | | | | | | | | | |
| DESCRIPTION | <p>With the <code>border()</code>, <code>wborder()</code>, and <code>box()</code> routines, a border is drawn around the edges of the window. The arguments and attributes are:</p> <hr/> <table border="0"> <tr> <td style="padding-right: 20px;"><i>ls</i></td> <td>left side of the border</td> </tr> <tr> <td><i>rs</i></td> <td>right side of the border</td> </tr> <tr> <td><i>ts</i></td> <td>top side of the border</td> </tr> <tr> <td><i>bs</i></td> <td>bottom side of the border</td> </tr> <tr> <td><i>tl</i></td> <td>top left-hand corner</td> </tr> <tr> <td><i>tr</i></td> <td>top right-hand corner</td> </tr> <tr> <td><i>bl</i></td> <td>bottom left-hand corner</td> </tr> <tr> <td><i>br</i></td> <td>bottom right-hand corner</td> </tr> </table> <hr/> <p>If any of these arguments is zero, then the following default values (defined in <code><curses.h></code>) are used respectively instead: <code>ACS_VLINE</code>, <code>ACS_VLINE</code>, <code>ACS_HLINE</code>, <code>ACS_HLINE</code>, <code>ACS_ULCORNER</code>, <code>ACS_URCORNER</code>, <code>ACS_BLCORNER</code>, <code>ACS_BRCORNER</code>.</p> <p><code>box(<i>win</i>, <i>verch</i>, <i>horch</i>)</code> is a shorthand for the following call:</p> <pre>wborder(<i>win</i>, <i>verch</i>, <i>verch</i>, <i>horch</i>, <i>horch</i>, 0, 0, 0, 0)</pre> <p><code>hline()</code> and <code>whline()</code> draw a horizontal (left to right) line using <i>ch</i> starting at the current cursor position in the window. The current cursor position is not changed. The line is at most <i>n</i> characters long, or as many as fit into the window.</p> | <i>ls</i> | left side of the border | <i>rs</i> | right side of the border | <i>ts</i> | top side of the border | <i>bs</i> | bottom side of the border | <i>tl</i> | top left-hand corner | <i>tr</i> | top right-hand corner | <i>bl</i> | bottom left-hand corner | <i>br</i> | bottom right-hand corner |
| <i>ls</i> | left side of the border | | | | | | | | | | | | | | | | |
| <i>rs</i> | right side of the border | | | | | | | | | | | | | | | | |
| <i>ts</i> | top side of the border | | | | | | | | | | | | | | | | |
| <i>bs</i> | bottom side of the border | | | | | | | | | | | | | | | | |
| <i>tl</i> | top left-hand corner | | | | | | | | | | | | | | | | |
| <i>tr</i> | top right-hand corner | | | | | | | | | | | | | | | | |
| <i>bl</i> | bottom left-hand corner | | | | | | | | | | | | | | | | |
| <i>br</i> | bottom right-hand corner | | | | | | | | | | | | | | | | |

whline(3CURSES)

vline() and wvline() draw a vertical (top to bottom) line using *ch* starting at the current cursor position in the window. The current cursor position is not changed. The line is at most *n* characters long, or as many as fit into the window.

RETURN VALUES All routines return the integer OK, or a non-negative integer if `immedok()` is set. See `curl_oulopl(3CURSES)`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO `curl_oulopl(3CURSES)`, `curls(3CURSES)`, `attributes(5)`

NOTES The header `<curls.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `border()` and `box()` may be macros.

| | |
|----------------------|--|
| NAME | hline, mvhline, mvvline, mvwhline, mvwvline, vline, whline, wvline – use single-byte characters (and renditions) to draw lines |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int hline(chtype ch, int n); int mvhline(int y, int x, chtype ch, int n); int mvvline(int y, int x, chtype ch, int n); int mvwhline(WINDOW *win, int y, int x, chtype ch, int n); int mvwvline(WINDOW *win, int y, int x, chtype ch, int n); int vline(chtype ch, int n); int whline(WINDOW *win, chtype ch, int n); int wvline(WINDOW *win, chtype ch, int n);</pre> |
| DESCRIPTION | <p>The <code>hline()</code>, <code>vline()</code>, <code>whline()</code>, <code>wvline()</code> functions draw a horizontal or vertical line, in either the window <code>stdscr</code> or <code>win</code> starting at the current cursor position. The line is drawn using the character <code>ch</code> and is a maximum of <code>n</code> positions long, or as many as will fit into the window. If <code>ch</code> is 0 (zero), the default horizontal or vertical character is used.</p> <p>The <code>mvhline()</code>, <code>mvvline()</code>, <code>mvwhline()</code>, <code>mvwvline()</code> functions are similar to the previous group of functions but the line begins at cursor position specified by <code>x</code> and <code>y</code>.</p> <p>The functions with names ending with <code>hline()</code> draw horizontal lines proceeding towards the last column of the same line. The functions with names ending with <code>vline()</code> draw vertical lines proceeding towards the last column of the same line.</p> <p>These functions do not change the position of the cursor.</p> |
| PARAMETERS | <p><code>ch</code> Is the character used to draw the line.</p> <p><code>n</code> Is the maximum number of characters in the line.</p> <p><code>y</code> Is the y (row) coordinate for the start of the line.</p> <p><code>x</code> Is the x (column) coordinate for the start of the line.</p> <p><code>win</code> Is a pointer to a window.</p> |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. |
| ERRORS | None |

whline(3XCURSES)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `border(3XCURSES)`, `border_set(3XCURSES)`, `hline_set(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| | |
|----------------------|---|
| NAME | hline_set, mvhline_set, mvvline_set, mvwhline_set, mvwvline_set, vline_set, whline_set, wvline_set – use complex characters (and renditions) to draw lines |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int hline_set(const cchar_t *ch, int n); int mvhline_set(int y, int x, const cchar_t *wch, int n); int mvvline_set(int y, int x, const cchar_t *wch, int n); int mvwhline_set(WINDOW *win, int y, int x, const cchar_t *wch, int n); int mvwvline_set(WINDOW *win, int y, int x, const cchar_t *wch, int n); int vline_set(const cchar_t *wch, int n); int whline_set(WINDOW *win, const cchar_t *wch, int n); int wvline_set(WINDOW *win, const cchar_t *wch, int n);</pre> |
| DESCRIPTION | <p>The <code>hline_set()</code>, <code>vline_set()</code>, <code>whline_set()</code>, <code>wvline_set()</code> functions draw a line, in either the window <code>stdscr</code> or <code>win</code> starting at the current cursor position. The line is drawn using the character <code>wch</code> and is a maximum of <code>n</code> positions long, or as many as will fit into the window. If <code>wch</code> is a null pointer, the default horizontal or vertical character is used.</p> <p>The <code>mvhline_set()</code>, <code>mvvline_set()</code>, <code>mvwhline_set()</code>, <code>mvwvline_set()</code> functions are similar to the previous group of functions but the line begins at cursor position specified by <code>x</code> and <code>y</code>.</p> <p>The functions with names ending with <code>hline_set()</code> draw horizontal lines proceeding towards the last column of the same line. The functions with names ending with <code>vline_set()</code> draw vertical lines proceeding towards the last column of the same line.</p> <p>These functions do not change the position of the cursor.</p> |
| PARAMETERS | <p><code>wch</code> Is the complex character used to draw the line.</p> <p><code>n</code> Is the maximum number of characters in the line.</p> <p><code>y</code> Is the y (row) coordinate for the start of the line.</p> <p><code>x</code> Is the x (column) coordinate for the start of the line.</p> <p><code>win</code> Is a pointer to a window.</p> |
| RETURN VALUES | On success, these functions return OK. Otherwise, they return ERR. |

whline_set(3XCURSES)

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `border(3XCURSES)`, `border_set(3XCURSES)`, `hline(3XCURSES)`,
`libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_inch, inch, winch, mvinch, mvwinch – get a character and its attributes from a curses window | | | | | | |
|--------------------|--|----------------|-------------------------------|--------------|--------------------------------|---------|--|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> ctype inch(void); ctype winch(WINDOW *win); ctype mvinch(int y, int x); ctype mvwinch(WINDOW *win, int y, int x);</pre> | | | | | | |
| DESCRIPTION | With these routines, the character, of type <code>ctype</code> , at the current position in the named window is returned. If any attributes are set for that position, their values are OR-ed into the value returned. Constants defined in <code><curses.h></code> can be used with the logical AND (<code>&</code>) operator to extract the character or attributes alone. | | | | | | |
| Attributes | <p>The following bit-masks can be AND-ed with characters returned by <code>winch()</code>.</p> <table> <tr> <td>A_CHARTEXT</td> <td>Bit-mask to extract character</td> </tr> <tr> <td>A_ATTRIBUTES</td> <td>Bit-mask to extract attributes</td> </tr> <tr> <td>A_COLOR</td> <td>Bit-mask to extract color-pair field information</td> </tr> </table> | A_CHARTEXT | Bit-mask to extract character | A_ATTRIBUTES | Bit-mask to extract attributes | A_COLOR | Bit-mask to extract color-pair field information |
| A_CHARTEXT | Bit-mask to extract character | | | | | | |
| A_ATTRIBUTES | Bit-mask to extract attributes | | | | | | |
| A_COLOR | Bit-mask to extract color-pair field information | | | | | | |
| ATTRIBUTES | See <code>attributes(5)</code> for descriptions of the following attributes: | | | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe | | |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | | | |
| MT-Level | Unsafe | | | | | | |
| SEE ALSO | <code>curses(3CURSES)</code> , <code>attributes(5)</code> | | | | | | |
| NOTES | <p>The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code>.</p> <p>Note that all of these routines may be macros.</p> | | | | | | |

winch(3XCURSES)

NAME | `inch`, `mvinch`, `mvwinch`, `winch` – return a single-byte character (with rendition)

SYNOPSIS |

```
cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [ library... ]

c89 [ flag... ] file... -lcurses [ library... ]

#include <curses.h>

ctype inch(void);

ctype mvinch(int y, int x);

ctype mvwinch(WINDOW *win, int y, int x);

ctype winch(WINDOW *win);
```

DESCRIPTION | The `inch()` and `winch()` functions return the `ctype` character located at the current cursor position of the `stdscr` window and window *win*, respectively. The `mvinch()` and `mvwinch()` functions return the `ctype` character located at the position indicated by the *x* (column) and *y* (row) parameters (the former in the `stdscr` window; the latter in window *win*).

The complete character/attribute pair will be returned. The character or attributes can be extracted by performing a bitwise AND on the returned value, using the constants `A_CHARTEXT`, `A_ATTRIBUTES`, and `A_COLOR`.

PARAMETERS | *y* | Is the *y* (row) coordinate of the position of the character to be returned.
x | Is the *x* (column) coordinate of the position of the character to be returned.
win | Is a pointer to the window that contains the character to be returned.

RETURN VALUES | On success, these functions return the specified character and rendition. Otherwise, they return `ERR`.

ERRORS | None.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO | `addch(3XCURSES)`, `attroff(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| NAME | curs_inchstr, inchstr, inchnstr, winchstr, winchnstr, mvinchstr, mvwinchnstr, mvwinchstr, mvwinchnstr – get a string of characters (and attributes) from a curses window | | | | |
|----------------------|--|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int inchstr(chtype *chstr); int inchnstr(chtype *chstr, int n); int winchstr(WINDOW *win, chtype *chstr); int winchnstr(WINDOW *win, chtype *chstr, int n); int mvinchstr(int y, int x, chtype *chstr); int mvwinchnstr(int y, int x, chtype *chstr, int n); int mvwinchstr(WINDOW *win, int y, int x, chtype *chstr); int mvwinchnstr(WINDOW *win, int y, int x, chtype *chstr, int n);</pre> | | | | |
| DESCRIPTION | With these routines, a string of type chtype, starting at the current cursor position in the named window and ending at the right margin of the window, is returned. The four functions with <i>n</i> as the last argument, return the string at most <i>n</i> characters long. Constants defined in <curses.h> can be used with the & (logical AND) operator to extract the character or the attribute alone from any position in the <i>chstr</i> (see curs_inch(3CURSES)). | | | | |
| RETURN VALUES | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curs_inch(3CURSES), curses(3CURSES), attributes(5) | | | | |
| NOTES | <p>The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.</p> <p>Note that all routines except winchnstr() may be macros.</p> | | | | |

winchnstr(3XCURSES)

| | | | | | | | | | |
|--------------------|---|--------------|---|----------|--|----------|--|----------|---|
| NAME | <code>inchnstr</code> , <code>inchstr</code> , <code>mvinchnstr</code> , <code>mvinchstr</code> , <code>mvwinchnstr</code> , <code>mvwinchstr</code> , <code>winchnstr</code> , <code>winchstr</code> – retrieve a single-byte character string (with rendition) | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int inchnstr(chtype *chstr, int n); int inchstr(chtype *chstr); int mvinchnstr(int y, int x, chtype *chstr, int n); int mvinchstr(int y, int x, chtype *chstr); int mvwinchnstr(WINDOW *win, int y, int x, chtype *chstr, int n); int mvwinchstr(WINDOW *win, int y, int x, chtype *chstr); int winchnstr(WINDOW *win, chtype *chstr, int n); int winchstr(WINDOW *win, chtype *chstr);</pre> | | | | | | | | |
| DESCRIPTION | <p>The <code>inchstr()</code> and <code>winchstr()</code> functions retrieve the character string (with rendition) starting at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively, and ending at the right margin. The <code>mvinchstr()</code> and <code>mvwinchstr()</code> functions retrieve the character string located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>inchnstr()</code>, <code>winchnstr()</code>, <code>mvinchnstr()</code>, and <code>mvwinchnstr()</code> functions retrieve at most <i>n</i> characters from the window <code>stdscr</code> and <i>win</i>, respectively. The former two functions retrieve the string, starting at the current cursor position; the latter two commands retrieve the string, starting at the position specified by the <i>x</i> and <i>y</i> parameters.</p> <p>All these functions store the retrieved character string in the object pointed to by <i>chstr</i>.</p> <p>The complete character/attribute pair is retrieved. The character or attributes can be extracted by performing a bitwise AND on the retrieved value, using the constants <code>A_CHARTEXT</code>, <code>A_ATTRIBUTES</code>, and <code>A_COLOR</code>. The character string can also be retrieved without attributes by using <code>instr(3XCURSES)</code> set of functions.</p> | | | | | | | | |
| PARAMETERS | <table><tr><td><i>chstr</i></td><td>Is a pointer to an object that can hold the retrieved character string.</td></tr><tr><td><i>n</i></td><td>Is the number of characters not to exceed when retrieving <i>chstr</i>.</td></tr><tr><td><i>y</i></td><td>Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved.</td></tr><tr><td><i>x</i></td><td>Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved.</td></tr></table> | <i>chstr</i> | Is a pointer to an object that can hold the retrieved character string. | <i>n</i> | Is the number of characters not to exceed when retrieving <i>chstr</i> . | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. |
| <i>chstr</i> | Is a pointer to an object that can hold the retrieved character string. | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when retrieving <i>chstr</i> . | | | | | | | | |
| <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | | | | | | | | |
| <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | | | | | | | | |

`winchnstr(3XCURSES)`

win Is a pointer to the window in which the string is to be retrieved.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `inch(3XCURSES)`, `innstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

winchstr(3CURSES)

| NAME | <code> curs_ inchstr, inchstr, inchnstr, winchstr, winchnstr, mvinchstr, mvinchnstr, mvwinchstr, mvwinchnstr </code> – get a string of characters (and attributes) from a curses window | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int inchstr(chtype *chstr); int inchnstr(chtype *chstr, int n); int winchstr(WINDOW *win, chtype *chstr); int winchnstr(WINDOW *win, chtype *chstr, int n); int mvinchstr(int y, int x, chtype *chstr); int mvinchnstr(int y, int x, chtype *chstr, int n); int mvwinchstr(WINDOW *win, int y, int x, chtype *chstr); int mvwinchnstr(WINDOW *win, int y, int x, chtype *chstr, int n);</pre> | | | | |
| DESCRIPTION | With these routines, a string of type <code>chtype</code> , starting at the current cursor position in the named window and ending at the right margin of the window, is returned. The four functions with <code>n</code> as the last argument, return the string at most <code>n</code> characters long. Constants defined in <code><curses.h></code> can be used with the <code>&</code> (logical AND) operator to extract the character or the attribute alone from any position in the <code>chstr</code> (see <code> curs_ inch(3CURSES)</code>). | | | | |
| RETURN VALUES | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion. | | | | |
| ATTRIBUTES | See <code> attributes(5)</code> for descriptions of the following attributes: | | | | |
| | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | <code> curs_ inch(3CURSES)</code> , <code> curses(3CURSES)</code> , <code> attributes(5)</code> | | | | |
| NOTES | The header <code><curses.h></code> automatically includes the headers <code><stdio.h></code> and <code><unctrl.h></code> . Note that all routines except <code> winchnstr()</code> may be macros. | | | | |

| | |
|--------------------|---|
| NAME | inchstr, inchstr, mvinchnstr, mvinchstr, mvwinchnstr, mvwinchstr, winchnstr, winchstr – retrieve a single-byte character string (with rendition) |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int inchstr(chtype *chstr, int n); int inchstr(chtype *chstr); int mvinchnstr(int y, int x, chtype *chstr, int n); int mvinchstr(int y, int x, chtype *chstr); int mvwinchnstr(WINDOW *win, int y, int x, chtype *chstr, int n); int mvwinchstr(WINDOW *win, int y, int x, chtype *chstr); int winchnstr(WINDOW *win, chtype *chstr, int n); int winchstr(WINDOW *win, chtype *chstr);</pre> |
| DESCRIPTION | <p>The <code>inchstr()</code> and <code>winchstr()</code> functions retrieve the character string (with rendition) starting at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively, and ending at the right margin. The <code>mvinchnstr()</code> and <code>mvwinchstr()</code> functions retrieve the character string located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>inchstr()</code>, <code>winchnstr()</code>, <code>mvinchnstr()</code>, and <code>mvwinchnstr()</code> functions retrieve at most <i>n</i> characters from the window <code>stdscr</code> and <i>win</i>, respectively. The former two functions retrieve the string, starting at the current cursor position; the latter two commands retrieve the string, starting at the position specified by the <i>x</i> and <i>y</i> parameters.</p> <p>All these functions store the retrieved character string in the object pointed to by <i>chstr</i>.</p> <p>The complete character/attribute pair is retrieved. The character or attributes can be extracted by performing a bitwise AND on the retrieved value, using the constants <code>A_CHARTEXT</code>, <code>A_ATTRIBUTES</code>, and <code>A_COLOR</code>. The character string can also be retrieved without attributes by using <code>instr(3XCURSES)</code> set of functions.</p> |
| PARAMETERS | <p><i>chstr</i> Is a pointer to an object that can hold the retrieved character string.</p> <p><i>n</i> Is the number of characters not to exceed when retrieving <i>chstr</i>.</p> <p><i>y</i> Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved.</p> <p><i>x</i> Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved.</p> |

winchstr(3XCURSES)

win Is a pointer to the window in which the string is to be retrieved.

RETURN VALUES On success, these functions return OK. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO inch(3XCURSES), innstr(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

| NAME | curs_instr, instr, innstr, winstr, winnstr, mvinstr, mvinnstr, mvwinstr, mvwinnstr – get a string of characters from a curses window | | | | |
|----------------------|---|----------------|-----------------|----------|--------|
| SYNOPSIS | <pre>cc [flag ...] file ... -lcurses [library ...] #include <curses.h> int instr(char *str); int innstr(char *str, int n); int winstr(WINDOW *win, char *str); int winnstr(WINDOW *win, char *str, int n); int mvinstr(int y, int x, char *str); int mvinnstr(int y, int x, char *str, int n); int mvwinstr(WINDOW *win, int y, int x, char *str); int mvwinnstr(WINDOW *win, int y, int x, char *str, int n);</pre> | | | | |
| DESCRIPTION | These routines return a string of characters in <i>str</i> , starting at the current cursor position in the named window and ending at the right margin of the window. Attributes are stripped from the characters. The four functions with <i>n</i> as the last argument return the string at most <i>n</i> characters long. | | | | |
| RETURN VALUES | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion. | | | | |
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes: | | | | |
| | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table> | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE | ATTRIBUTE VALUE | | | | |
| MT-Level | Unsafe | | | | |
| SEE ALSO | curses(3CURSES), attributes(5) | | | | |
| NOTES | <p>The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.</p> <p>Note that all routines except winnstr() may be macros.</p> | | | | |

winnstr(3XCURSES)

| | | | | | | | | | | | |
|--------------------|--|------------|---|----------|--|----------|--|----------|---|------------|--|
| NAME | <code>innstr</code> , <code>instr</code> , <code>mvinnstr</code> , <code>mvinstr</code> , <code>mvwinnstr</code> , <code>mvwinstr</code> , <code>winnstr</code> , <code>winstr</code> – retrieve a multibyte character string (without rendition) | | | | | | | | | | |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int innstr(char *str, int n); int instr(char *str); int mvinnstr(int y, int x, char *str, int n); int mvinstr(int y, int x, char *str); int mvwinnstr(WINDOW *win, int y, int x, char *str, int n); int mvwinstr(WINDOW *win, int y, int x, char *str); int winstr(WINDOW *win, char *str); int winnstr(WINDOW *win, char *str, int n);</pre> | | | | | | | | | | |
| PARAMETERS | <table><tr><td><i>str</i></td><td>Is a pointer to an object that can hold the retrieved multibyte character string.</td></tr><tr><td><i>n</i></td><td>Is the number of characters not to exceed when retrieving <i>str</i>.</td></tr><tr><td><i>y</i></td><td>Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved.</td></tr><tr><td><i>x</i></td><td>Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved.</td></tr><tr><td><i>win</i></td><td>Is a pointer to the window in which the string is to be retrieved.</td></tr></table> | <i>str</i> | Is a pointer to an object that can hold the retrieved multibyte character string. | <i>n</i> | Is the number of characters not to exceed when retrieving <i>str</i> . | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | <i>win</i> | Is a pointer to the window in which the string is to be retrieved. |
| <i>str</i> | Is a pointer to an object that can hold the retrieved multibyte character string. | | | | | | | | | | |
| <i>n</i> | Is the number of characters not to exceed when retrieving <i>str</i> . | | | | | | | | | | |
| <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | | | | | | | | | | |
| <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | | | | | | | | | | |
| <i>win</i> | Is a pointer to the window in which the string is to be retrieved. | | | | | | | | | | |
| DESCRIPTION | <p>The <code>instr()</code> and <code>winstr()</code> functions retrieve a multibyte character string (without attributes) starting at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively, and ending at the right margin. The <code>mvinstr()</code> and <code>mvwinstr()</code> functions retrieve a multibyte character string located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>innstr()</code>, <code>winnstr()</code>, <code>mvinnstr()</code>, and <code>mvwinnstr()</code> functions retrieve at most <i>n</i> characters from the window <code>stdscr</code> and <i>win</i>, respectively. The former two functions retrieve the string starting at the current cursor position; the latter two commands return the string, starting at the position specified by the <i>x</i> and <i>y</i> parameters.</p> | | | | | | | | | | |

winnstr(3XCURSES)

All these functions store the retrieved string in the object pointed to by *str*. They only store complete multibyte characters. If the area pointed to by *str* is not large enough to hold at least one character, these functions fail.

Only the character portion of the character/rendition pair is returned. To return the complete character/rendition pair, use `winchstr()`.

ERRORS OK Successful completion.

ERR An error occurred.

USAGE All functions except `winnstr()` may be macros.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `inch(3XCURSES)`, `inchstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

winnwstr(3CURSES)

NAME | curs_inwstr, inwstr, innwstr, winwstr, winnwstr, mvinwstr, mvinnwstr, mvwinwstr, mvwinnwstr – get a string of wchar_t characters from a curses window

SYNOPSIS | `cc [flag ...] file ... -lcurses [library ..]`
| `#include <curses.h>`
|
| `int inwstr(wchar_t *wstr);`
| `int innwstr(wchar_t *wstr, int n);`
| `int winwstr(WINDOW *win, wchar_t *wstr);`
| `int winnwstr(WINDOW *win, wchar_t *wstr, int n);`
| `int mvinwstr(int y, int x, wchar_t *wstr);`
| `int mvinnwstr(int y, int x, wchar_t *wstr, int n);`
| `int mvwinwstr(WINDOW *win, int y, int x, wchar_t *wstr);`
| `int mvwinnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);`

DESCRIPTION | These routines return the string of wchar_t characters in wstr starting at the current cursor position in the named window and ending at the right margin of the window. Attributes are stripped from the characters. The four functions with n as the last argument return the string at most n wchar_t characters long.

RETURN VALUES | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level | Unsafe |

SEE ALSO | curses(3CURSES), attributes(5)

NOTES | The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <wctype.h>.

Note that all routines except winnwstr() may be macros.

| | |
|--------------------|---|
| NAME | innwstr, inwstr, mvinnwstr, mvinwstr, mvwinnwstr, mvwinwstr, winnwstr, winwstr – retrieve a wide character string (without rendition) |
| SYNOPSIS | <pre>cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [library...] c89 [flag...] file... -lcurses [library...] #include <curses.h> int innwstr(wchar_t *wstr, int n); int inwstr(wchar_t *wstr); int mvinnwstr(int y, int x, wchar_t *wstr, int n); int mvinwstr(int y, int x, wchar_t *wstr); int mvwinnwstr(WINDOW*win, int y, int x, wchar_t *wstr, int n); int mvwinwstr(WINDOW*win, int y, int x, wchar_t *wstr); int winwstr(WINDOW*win, wchar_t *wstr); int winnwstr(WINDOW*win, wchar_t *wstr, int n);</pre> |
| PARAMETERS | <p><i>wstr</i> Is a pointer to an object that can hold the retrieved multibyte character string.</p> <p><i>n</i> Is the number of characters not to exceed when retrieving <i>wstr</i>.</p> <p><i>y</i> Is the y (row) coordinate of the starting position of the string to be retrieved.</p> <p><i>x</i> Is the x (column) coordinate of the starting position of the string to be retrieved.</p> <p><i>win</i> Is a pointer to the window in which the string is to be retrieved.</p> |
| DESCRIPTION | <p>The <code>inwstr()</code> and <code>winwstr()</code> functions retrieve a wide character string (without attributes) starting at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively, and ending at the right margin. The <code>mvinwstr()</code> and <code>mvwinwstr()</code> functions retrieve a wide character string located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>innwstr()</code>, <code>winnwstr()</code>, <code>mvinnwstr()</code>, and <code>mvwinnwstr()</code> functions retrieve at most <i>n</i> characters from the window <code>stdscr</code> and <i>win</i>, respectively. The former two functions retrieve the string starting at the current cursor position; the latter two commands return the string, starting at the position specified by the <i>x</i> and <i>y</i> parameters.</p> <p>All these functions store the retrieved string in the object pointed to by <i>wstr</i>. They only store complete wide characters. If the area pointed to by <i>wstr</i> is not large enough to hold at least one character, these functions fail.</p> |

winnwstr(3XCURSES)

Only the character portion of the character/rendition pair is returned. To return the complete character/rendition pair, use `win_wchstr(3XCURSES)`.

RETURN VALUES On success, the `inwstr()`, `mvinwstr()`, `mvwinwstr()`, and `winnwstr()` functions return OK. Otherwise, they return ERR.

On success, the `innwstr()`, `mvinnwstr()`, `mvwinnwstr()`, and `winnwstr()` functions return the number of characters read into the string. Otherwise, they return ERR.

ERRORS None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard |
| MT-Level | Unsafe |

SEE ALSO `in_wch(3XCURSES)`, `in_wchnstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

- NAME** curs_insch, insch, winsch, mvinsch, mvwinsch – insert a character before the character under the cursor in a curses window
- SYNOPSIS**
- ```
cc [flag ...] file... -lcurses [library ...]
#include <curses.h>

int insch(chtype ch);
int winsch(WINDOW *win, chtype ch);
int mvinsch(int y, int x, chtype ch);
int mvwinsch(WINDOW *win, int y, int x, chtype ch);
```
- DESCRIPTION** With these routines, the character *ch* is inserted before the character under the cursor. All characters to the right of the cursor are moved one space to the right, with the possibility of the rightmost character on the line being lost. The cursor position does not change (after moving to *y*, *x*, if specified). (This does not imply use of the hardware insert character feature.)
- RETURN VALUES** All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.
- ATTRIBUTES** See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level       | Unsafe          |

- SEE ALSO** curses(3CURSES), attributes(5)
- NOTES** The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.
- Note that `insch()`, `mvinsch()`, and `mvwinsch()` may be macros.

winsch(3XCURSES)

**NAME** | insch, winsch, mvinsch, mvwinsch – insert a character

**SYNOPSIS** | `cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \`  
`-R /usr/xpg4/lib -lcurses [ library... ]`

**c89** | `[ flag... ] file... -lcurses [ library... ]`

```
#include <curses.h>
int insch(chtype ch);
int mvinsch(int y, int x, chtype ch);
int mvwinsch(WINDOW *win, int y, int x, chtype ch);
int winsch(WINDOW *win, chtype ch);
```

**PARAMETERS**

|            |                                                                      |
|------------|----------------------------------------------------------------------|
| <i>ch</i>  | Is the character to be inserted.                                     |
| <i>y</i>   | Is the y (row) coordinate of the position of the character.          |
| <i>x</i>   | Is the x (column) coordinate of the position of the character.       |
| <i>win</i> | Is a pointer to the window in which the character is to be inserted. |

**DESCRIPTION** | These functions insert the character and rendition from *ch* into the current or specified window at the current or specified position.

These functions do not perform wrapping and do not advance the cursor position. These functions perform special-character processing, with the exception that if a newline is inserted into the last line of a window and scrolling is not enabled, the behavior is unspecified.

**RETURN VALUES** | Upon successful completion, these functions return OK. Otherwise, they return ERR.

**ERRORS** | No errors are defined.

**USAGE** | These functions are only guaranteed to operate reliably on character sets in which each character fits into a single byte, whose attributes can be expressed using only constants with the A\_ prefix.

**ATTRIBUTES** | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard        |
| MT-Level            | Unsafe          |

**SEE ALSO** | ins\_wch(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

| <b>NAME</b>          | curs_deleteln, deleteln, wdeleteln, insdelln, winsdelln, insertln, winsertln – delete and insert lines in a curses window                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                |                 |          |        |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|----------|--------|
| <b>SYNOPSIS</b>      | <pre>cc [ flag ... ] file ... -lcurses [ library ... ] #include &lt;curses.h&gt;  int deleteln(void); int wdeleteln(WINDOW *win); int insdelln(int n); int winsdelln(WINDOW *win, int n); int insertln(void); int winsertln(WINDOW *win);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                |                 |          |        |
| <b>DESCRIPTION</b>   | <p>With the <code>deleteln()</code> and <code>wdeleteln()</code> routines, the line under the cursor in the window is deleted; all lines below the current line are moved up one line. The bottom line of the window is cleared. The cursor position does not change. This does not imply use of a hardware delete line feature.</p> <p>With the <code>insdelln()</code> and <code>winsdelln()</code> routines, for positive <math>n</math>, insert <math>n</math> lines into the specified window above the current line. The <math>n</math> bottom lines are lost. For negative <math>n</math>, delete <math>n</math> lines (starting with the one under the cursor), and move the remaining lines up. The bottom <math>n</math> lines are cleared. The current cursor position remains the same.</p> <p>With the <code>insertln()</code> and <code>winsertln()</code> routines, a blank line is inserted above the current line and the bottom line is lost. This does not imply use of a hardware insert line feature.</p> |                |                 |          |        |
| <b>RETURN VALUES</b> | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                |                 |          |        |
| <b>ATTRIBUTES</b>    | See <code>attributes(5)</code> for descriptions of the following attributes:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                |                 |          |        |
|                      | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">MT-Level</td> <td style="text-align: center;">Unsafe</td> </tr> </tbody> </table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE       | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                |                 |          |        |
| MT-Level             | Unsafe                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                |                 |          |        |
| <b>SEE ALSO</b>      | <code>curses(3CURSES)</code> , <code>attributes(5)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                |                 |          |        |
| <b>NOTES</b>         | <p>The header <code>&lt;curses.h&gt;</code> automatically includes the headers <code>&lt;stdio.h&gt;</code> and <code>&lt;unctrl.h&gt;</code>.</p> <p>Note that all but <code>winsdelln()</code> may be macros.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                |                 |          |        |

## winsdelln(3XCURSES)

**NAME** | insdelln, winsdelln – insert/delete lines to/from the window

**SYNOPSIS** | 

```
cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [library...]

c89 [flag...] file... -lcurses [library...]

#include <curses.h>

int insdelln(int n);

int winsdelln(WINDOW *win, int n);
```

**PARAMETERS** | *n* | Is the number of lines to insert or delete (positive *n* inserts; negative *n* deletes).

| *win* | Is a pointer to the window in which to insert or delete a line.

**DESCRIPTION** | The insdelln() and winsdelln() functions insert or delete blank lines in stdscr or win, respectively. When *n* is positive, *n* lines are added before the current line and the bottom *n* lines are lost; when *n* is negative, *n* lines are deleted starting with the current line, the remaining lines are moved up, and the bottom *n* lines are cleared. The position of the cursor does not change.

**RETURN VALUES** | On success, these functions return OK. Otherwise, they return ERR.

**ERRORS** | None.

**ATTRIBUTES** | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard        |
| MT-Level            | Unsafe          |

**SEE ALSO** | deleteln(3XCURSES), insertln(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

| <b>NAME</b>          | curs_deleteln, deleteln, wdeleteln, insdelln, winsdelln, insertln, winsertln – delete and insert lines in a curses window                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                |                 |          |        |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|----------|--------|
| <b>SYNOPSIS</b>      | <pre>cc [ flag ... ] file ... -lcurses [ library ... ] #include &lt;curses.h&gt;  int deleteln(void); int wdeleteln(WINDOW *win); int insdelln(int n); int winsdelln(WINDOW *win, int n); int insertln(void); int winsertln(WINDOW *win);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                |                 |          |        |
| <b>DESCRIPTION</b>   | <p>With the <code>deleteln()</code> and <code>wdeleteln()</code> routines, the line under the cursor in the window is deleted; all lines below the current line are moved up one line. The bottom line of the window is cleared. The cursor position does not change. This does not imply use of a hardware delete line feature.</p> <p>With the <code>insdelln()</code> and <code>winsdelln()</code> routines, for positive <math>n</math>, insert <math>n</math> lines into the specified window above the current line. The <math>n</math> bottom lines are lost. For negative <math>n</math>, delete <math>n</math> lines (starting with the one under the cursor), and move the remaining lines up. The bottom <math>n</math> lines are cleared. The current cursor position remains the same.</p> <p>With the <code>insertln()</code> and <code>winsertln()</code> routines, a blank line is inserted above the current line and the bottom line is lost. This does not imply use of a hardware insert line feature.</p> |                |                 |          |        |
| <b>RETURN VALUES</b> | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                |                 |          |        |
| <b>ATTRIBUTES</b>    | See <code>attributes(5)</code> for descriptions of the following attributes:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                |                 |          |        |
|                      | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE       | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                |                 |          |        |
| MT-Level             | Unsafe                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                |                 |          |        |
| <b>SEE ALSO</b>      | <code>curses(3CURSES)</code> , <code>attributes(5)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                |                 |          |        |
| <b>NOTES</b>         | <p>The header <code>&lt;curses.h&gt;</code> automatically includes the headers <code>&lt;stdio.h&gt;</code> and <code>&lt;unctrl.h&gt;</code>.</p> <p>Note that all but <code>winsdelln()</code> may be macros.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                |                 |          |        |

## winsertln(3XCURSES)

**NAME** insertln, winsertln – insert a line in a window

**SYNOPSIS**

```
cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [library...]

c89 [flag...] file... -lcurses [library...]

#include <curses.h>

int insertln(void);

int winsertln(WINDOW *win);
```

**PARAMETERS** *win* Is a pointer to the window in which to insert the line.

**DESCRIPTION** The insertln() and winsertln() functions insert a blank line before the current line in stdscr or *win*, respectively. The new line becomes the current line. The current line and all lines after it in the window are moved down one line. The bottom line in the window is discarded.

**RETURN VALUES** On success, these functions return OK. Otherwise, they return ERR.

**ERRORS** None.

**ATTRIBUTES** See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard        |
| MT-Level            | Unsafe          |

**SEE ALSO** bkgdset(3XCURSES), deleteln(3XCURSES), insdelln(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

| <b>NAME</b>          | curs_insstr, insstr, insnstr, winsstr, winsnstr, mvinsstr, mvinsnstr, mvwinsstr, mvwinsnstr – insert string before character under the cursor in a curses window                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                |                 |          |        |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|----------|--------|
| <b>SYNOPSIS</b>      | <pre>cc [ flag ... ] file ... -lcurses [ library ... ] #include &lt;curses.h&gt;  int insstr(char *str); int insnstr(char *str, int n); int winsstr(WINDOW *win, char *str); int winsnstr(WINDOW *win, char *str, int n); int mvinsstr(int y, int x, char *str); int mvinsnstr(int y, int x, char *str, int n); int mvwinsstr(WINDOW *win, int y, int x, char *str); int mvwinsnstr(WINDOW *win, int y, int x, char *str, int n);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                |                 |          |        |
| <b>DESCRIPTION</b>   | <p>With these routines, a character string (as many characters as will fit on the line) is inserted before the character under the cursor. All characters to the right of the cursor are moved to the right, with the possibility of the rightmost characters on the line being lost. The cursor position does not change (after moving to <i>y, x</i>, if specified). (This does not imply use of the hardware insert character feature.) The four routines with <i>n</i> as the last argument insert at most <i>n</i> characters. If <i>n</i> ≤ 0, then the entire string is inserted.</p> <p>If a character in <i>str</i> is a tab, newline, carriage return or backspace, the cursor is moved appropriately within the window. A newline also does a <code>clrtoeol()</code> before moving. Tabs are considered to be at every eighth column. If a character in <i>str</i> is another control character, it is drawn in the <code>^X</code> notation. Calling <code>winch()</code> after adding a control character (and moving to it, if necessary) does not return the control character, but instead returns the representation of the control character.</p> |                |                 |          |        |
| <b>RETURN VALUES</b> | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                |                 |          |        |
| <b>ATTRIBUTES</b>    | See <code>attributes(5)</code> for descriptions of the following attributes:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                |                 |          |        |
|                      | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">MT-Level</td> <td style="text-align: center;">Unsafe</td> </tr> </tbody> </table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE       | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                |                 |          |        |
| MT-Level             | Unsafe                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                |                 |          |        |
| <b>SEE ALSO</b>      | <code>curs_clear(3CURSES)</code> , <code>curs_inch(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                |                 |          |        |
| <b>NOTES</b>         | The header <code>&lt;curses.h&gt;</code> automatically includes the headers <code>&lt;stdio.h&gt;</code> and <code>&lt;unctrl.h&gt;</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                |                 |          |        |

winsnstr(3CURSES)

Note that all but `winsnstr()` may be macros.

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |            |                                            |          |                                                                                                                                  |          |                                                                          |          |                                                                             |            |                                                                   |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|--------------------------------------------|----------|----------------------------------------------------------------------------------------------------------------------------------|----------|--------------------------------------------------------------------------|----------|-----------------------------------------------------------------------------|------------|-------------------------------------------------------------------|
| <b>NAME</b>        | insnstr, insstr, mvinsnstr, mvinsstr, mvwinsnstr, mvwinsstr, winsnstr, winsstr – insert a multibyte character string                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |            |                                            |          |                                                                                                                                  |          |                                                                          |          |                                                                             |            |                                                                   |
| <b>SYNOPSIS</b>    | <pre>cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [ library... ]  c89 [ flag... ] file... -lcurses [ library... ]  #include &lt;curses.h&gt;  int insnstr(const char *str, int n); int insstr(const char *str); int mvinsnstr(int y, int x, const char *str, int n); int mvinsstr(int y, int x, const char *str); int mvwinsnstr(WINDOW *win, int y, int x, const char *str, int n); int mvwinsstr(WINDOW *win, int y, int x, const char *str); int winsnstr(WINDOW *win, const char *str, int n); int winsstr(WINDOW *win, const char *str);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |            |                                            |          |                                                                                                                                  |          |                                                                          |          |                                                                             |            |                                                                   |
| <b>PARAMETERS</b>  | <table border="0"> <tr> <td style="padding-right: 20px;"><i>str</i></td> <td>Is a pointer to the string to be inserted.</td> </tr> <tr> <td><i>n</i></td> <td>Is the number of characters not to exceed when inserting <i>str</i>. If <i>n</i> is less than 1, the entire string is inserted.</td> </tr> <tr> <td><i>y</i></td> <td>Is the <i>y</i> (row) coordinate of the starting position of the string.</td> </tr> <tr> <td><i>x</i></td> <td>Is the <i>x</i> (column) coordinate of the starting position of the string.</td> </tr> <tr> <td><i>win</i></td> <td>Is a pointer to the window in which the string is to be inserted.</td> </tr> </table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | <i>str</i> | Is a pointer to the string to be inserted. | <i>n</i> | Is the number of characters not to exceed when inserting <i>str</i> . If <i>n</i> is less than 1, the entire string is inserted. | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string. | <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string. | <i>win</i> | Is a pointer to the window in which the string is to be inserted. |
| <i>str</i>         | Is a pointer to the string to be inserted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |            |                                            |          |                                                                                                                                  |          |                                                                          |          |                                                                             |            |                                                                   |
| <i>n</i>           | Is the number of characters not to exceed when inserting <i>str</i> . If <i>n</i> is less than 1, the entire string is inserted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |            |                                            |          |                                                                                                                                  |          |                                                                          |          |                                                                             |            |                                                                   |
| <i>y</i>           | Is the <i>y</i> (row) coordinate of the starting position of the string.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |            |                                            |          |                                                                                                                                  |          |                                                                          |          |                                                                             |            |                                                                   |
| <i>x</i>           | Is the <i>x</i> (column) coordinate of the starting position of the string.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |            |                                            |          |                                                                                                                                  |          |                                                                          |          |                                                                             |            |                                                                   |
| <i>win</i>         | Is a pointer to the window in which the string is to be inserted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |            |                                            |          |                                                                                                                                  |          |                                                                          |          |                                                                             |            |                                                                   |
| <b>DESCRIPTION</b> | <p>The <code>insstr()</code> function inserts <i>str</i> at the current cursor position of the <code>stdscr</code> window. The <code>winsstr()</code> function performs the identical action, but in window <i>win</i>. The <code>mvinsstr()</code> and <code>mvwinsstr()</code> functions insert the character string at the starting position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <i>win</i>).</p> <p>The <code>insnstr()</code>, <code>winsnstr()</code>, <code>mvinsnstr()</code>, and <code>mvwinsnstr()</code> functions insert <i>n</i> characters to the window or as many as will fit on the line. If <i>n</i> is less than 1, the entire string is inserted or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <i>x</i> and <i>y</i> parameters.</p> <p>All characters to the right of inserted characters are moved to the right. Characters that don't fit on the current line are discarded. The cursor is left at the point of insertion.</p> |            |                                            |          |                                                                                                                                  |          |                                                                          |          |                                                                             |            |                                                                   |

winsnstr(3XCURSES)

If a character in *str* is a newline, carriage return, backspace, or tab, the cursor is moved appropriately. The cursor is moved to the next tab stop for each tab character (by default, tabs are eight characters apart). If the character is a control character other than those previously mentioned, the character is inserted using  $\^x$  notation, where *x* is a printable character. `clrtoeol(3XCURSES)` is automatically done before a newline.

**RETURN VALUES** On success, these functions return OK. Otherwise, they return ERR.

**ERRORS** None.

**ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard        |
| MT-Level            | Unsafe          |

**SEE ALSO** `addchstr(3XCURSES)`, `addstr(3XCURSES)`, `clrtoeol(3XCURSES)`, `ins_nwstr(3XCURSES)`, `insch(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| <b>NAME</b>         | curs_inswstr, inswstr, insnwstr, winswstr, winsnwstr, mvinswstr, mvinsnwstr, mvwinswstr, mvwinsnwstr – insert wchar_t string before character under the cursor in a curses window                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                |                 |          |        |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|----------|--------|
| <b>SYNOPSIS</b>     | <pre>cc [ flag ... ] file ... -lcurses [ library .. ] #include &lt;curses.h&gt;  int inswstr(wchar_t *wstr); int insnwstr(wchar_t *wstr, int n); int winswstr(WINDOW *win, wchar_t *wstr); int winsnwstr(WINDOW *win, wchar_t *wstr, int n); int mvinswstr(int y, int x, wchar_t *wstr); int mvinsnwstr(int y, int x, wchar_t *wstr, int n); int mvwinswstr(WINDOW *win, int y, int x, wchar_t *wstr); int mvwinsnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                |                 |          |        |
| <b>DESCRIPTION</b>  | <p>These routines insert a wchar_t character string (as many wchar_t characters as will fit on the line) before the character under the cursor. All characters to the right of the cursor are moved to the right, with the possibility of the rightmost characters on the line being lost. The cursor position does not change (after moving to <i>y, x</i>, if specified). (This does not imply use of the hardware insert character feature.) The four routines with <i>n</i> as the last argument insert at most <i>n</i> wchar_t characters. If <i>n</i> ≤ 0, then the entire string is inserted.</p> <p>If a character in <i>wstr</i> is a tab, newline, carriage return, or backspace, the cursor is moved appropriately within the window. A newline also does a clrtoeol(3CURSES) before moving. Tabs are considered to be at every eighth column. If a character in <i>wstr</i> is another control character, it is drawn in the ^X notation. Calling winwch(3CURSES) after adding a control character (and moving to it, if necessary) does not return the control character, but instead returns the representation of the control character.</p> |                |                 |          |        |
| <b>RETURN VALUE</b> | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                |                 |          |        |
| <b>ATTRIBUTES</b>   | See attributes(5) for descriptions of the following attributes:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                |                 |          |        |
|                     | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE      | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                |                 |          |        |
| MT-Level            | Unsafe                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                |                 |          |        |
| <b>SEE ALSO</b>     | clrtoeol(3CURSES), curses(3CURSES), winwch(3CURSES), attributes(5)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                |                 |          |        |
| <b>NOTES</b>        | The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <wchar.h>.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                |                 |          |        |

winsnwstr(3CURSES)

Note that all but `winsnwstr()` may be macros.

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |             |                                            |          |                                                                                                                                   |          |                                                                          |          |                                                                             |            |                                                                   |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|--------------------------------------------|----------|-----------------------------------------------------------------------------------------------------------------------------------|----------|--------------------------------------------------------------------------|----------|-----------------------------------------------------------------------------|------------|-------------------------------------------------------------------|
| <b>NAME</b>        | ins_nwstr, ins_wstr, mvins_nwstr, mvins_wstr, mvwins_nwstr, mvwins_wstr, wins_nwstr, wins_wstr – insert a wide character string                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |             |                                            |          |                                                                                                                                   |          |                                                                          |          |                                                                             |            |                                                                   |
| <b>SYNOPSIS</b>    | <pre>cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [ library... ]  c89 [ flag... ] file... -lcurses [ library... ]  #include &lt;curses.h&gt;  int ins_nwstr(const wchar_t *wstr, int n); int ins_wstr(const wchar_t *wstr); int mvins_nwstr(int y, int x, const wchar_t *wstr, int n); int mvins_wstr(int y, int x, const wchar_t *wstr); int mvwins_nwstr(WINDOW *win, int y, int x, const wchar_t *wstr, int n); int mvwins_wstr(WINDOW *win, int y, int x, const wchar_t *wstr); int wins_nwstr(WINDOW *win, const wchar_t *wstr, int n); int wins_wstr(WINDOW *win, const wchar_t *wstr);</pre>                                                                                                                                                                                                                                                                                                                                                       |             |                                            |          |                                                                                                                                   |          |                                                                          |          |                                                                             |            |                                                                   |
| <b>PARAMETERS</b>  | <table border="0"> <tr> <td style="padding-right: 20px;"><i>wstr</i></td> <td>Is a pointer to the string to be inserted.</td> </tr> <tr> <td><i>n</i></td> <td>Is the number of characters not to exceed when inserting <i>wstr</i>. If <i>n</i> is less than 1, the entire string is inserted.</td> </tr> <tr> <td><i>y</i></td> <td>Is the <i>y</i> (row) coordinate of the starting position of the string.</td> </tr> <tr> <td><i>x</i></td> <td>Is the <i>x</i> (column) coordinate of the starting position of the string.</td> </tr> <tr> <td><i>win</i></td> <td>Is a pointer to the window in which the string is to be inserted.</td> </tr> </table>                                                                                                                                                                                                                                                                                                                                        | <i>wstr</i> | Is a pointer to the string to be inserted. | <i>n</i> | Is the number of characters not to exceed when inserting <i>wstr</i> . If <i>n</i> is less than 1, the entire string is inserted. | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string. | <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string. | <i>win</i> | Is a pointer to the window in which the string is to be inserted. |
| <i>wstr</i>        | Is a pointer to the string to be inserted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |                                            |          |                                                                                                                                   |          |                                                                          |          |                                                                             |            |                                                                   |
| <i>n</i>           | Is the number of characters not to exceed when inserting <i>wstr</i> . If <i>n</i> is less than 1, the entire string is inserted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |             |                                            |          |                                                                                                                                   |          |                                                                          |          |                                                                             |            |                                                                   |
| <i>y</i>           | Is the <i>y</i> (row) coordinate of the starting position of the string.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |             |                                            |          |                                                                                                                                   |          |                                                                          |          |                                                                             |            |                                                                   |
| <i>x</i>           | Is the <i>x</i> (column) coordinate of the starting position of the string.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |             |                                            |          |                                                                                                                                   |          |                                                                          |          |                                                                             |            |                                                                   |
| <i>win</i>         | Is a pointer to the window in which the string is to be inserted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |             |                                            |          |                                                                                                                                   |          |                                                                          |          |                                                                             |            |                                                                   |
| <b>DESCRIPTION</b> | <p>The <code>ins_wstr()</code> function inserts <i>wstr</i> at the current cursor position of the <code>stdscr</code> window. The <code>wins_wstr()</code> function performs the identical action, but in window <i>win</i>. The <code>mvins_wstr()</code> and <code>mvwins_wstr()</code> functions insert <i>wstr</i> string at the starting position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>ins_nwstr()</code>, <code>wins_nwstr()</code>, <code>mvins_nwstr()</code>, and <code>mvwins_nwstr()</code> functions insert <i>n</i> characters to the window or as many as will fit on the line. If <i>n</i> is less than 1, the entire string is inserted or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <i>x</i> and <i>y</i> parameters.</p> |             |                                            |          |                                                                                                                                   |          |                                                                          |          |                                                                             |            |                                                                   |

wins\_nwstr(3XCURSES)

All characters to the right of inserted characters are moved to the right. Characters that don't fit on the current line are discarded. The cursor is left at the point of insertion.

If a character in *wstr* is a newline, carriage return, backspace, or tab, the cursor is moved appropriately. The cursor is moved to the next tab stop for each tab character (by default, tabs are eight characters apart). If the character is a control character other than those previously mentioned, the character is inserted using  $\hat{x}$  notation, where  $x$  is a printable character. `clrtoeol(3XCURSES)` is automatically done before a newline.

**RETURN VALUES** On success, these functions return OK. Otherwise, they return ERR.

**ERRORS** None.

**ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard        |
| MT-Level            | Unsafe          |

**SEE ALSO** `add_wchnstr(3XCURSES)`, `addnwstr(3XCURSES)`, `clrtoeol(3XCURSES)`, `ins_wch(3XCURSES)`, `insnstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| <b>NAME</b>          | curs_insstr, insstr, insnstr, winsstr, winsnstr, mvinsstr, mvinsnstr, mvwinsstr, mvwinsnstr – insert string before character under the cursor in a curses window                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                |                 |          |        |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|----------|--------|
| <b>SYNOPSIS</b>      | <pre>cc [ flag ... ] file ... -lcurses [ library ... ] #include &lt;curses.h&gt;  int insstr(char *str); int insnstr(char *str, int n); int winsstr(WINDOW *win, char *str); int winsnstr(WINDOW *win, char *str, int n); int mvinsstr(int y, int x, char *str); int mvinsnstr(int y, int x, char *str, int n); int mvwinsstr(WINDOW *win, int y, int x, char *str); int mvwinsnstr(WINDOW *win, int y, int x, char *str, int n);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                |                 |          |        |
| <b>DESCRIPTION</b>   | <p>With these routines, a character string (as many characters as will fit on the line) is inserted before the character under the cursor. All characters to the right of the cursor are moved to the right, with the possibility of the rightmost characters on the line being lost. The cursor position does not change (after moving to <i>y, x</i>, if specified). (This does not imply use of the hardware insert character feature.) The four routines with <i>n</i> as the last argument insert at most <i>n</i> characters. If <i>n</i> ≤ 0, then the entire string is inserted.</p> <p>If a character in <i>str</i> is a tab, newline, carriage return or backspace, the cursor is moved appropriately within the window. A newline also does a <code>clrtoeol()</code> before moving. Tabs are considered to be at every eighth column. If a character in <i>str</i> is another control character, it is drawn in the <code>^X</code> notation. Calling <code>winch()</code> after adding a control character (and moving to it, if necessary) does not return the control character, but instead returns the representation of the control character.</p> |                |                 |          |        |
| <b>RETURN VALUES</b> | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                |                 |          |        |
| <b>ATTRIBUTES</b>    | See <code>attributes(5)</code> for descriptions of the following attributes:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                |                 |          |        |
|                      | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE       | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                |                 |          |        |
| MT-Level             | Unsafe                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                |                 |          |        |
| <b>SEE ALSO</b>      | <code>curs_clear(3CURSES)</code> , <code>curs_inch(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                |                 |          |        |
| <b>NOTES</b>         | The header <code>&lt;curses.h&gt;</code> automatically includes the headers <code>&lt;stdio.h&gt;</code> and <code>&lt;unctrl.h&gt;</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                |                 |          |        |

winsstr(3CURSES)

Note that all but `winsnstr()` may be macros.

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |            |                                            |          |                                                                                                                                  |          |                                                                          |          |                                                                             |            |                                                                   |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|--------------------------------------------|----------|----------------------------------------------------------------------------------------------------------------------------------|----------|--------------------------------------------------------------------------|----------|-----------------------------------------------------------------------------|------------|-------------------------------------------------------------------|
| <b>NAME</b>        | insnstr, insstr, mvinsnstr, mvinsstr, mvwinsnstr, mvwinsstr, winsnstr, winsstr – insert a multibyte character string                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |            |                                            |          |                                                                                                                                  |          |                                                                          |          |                                                                             |            |                                                                   |
| <b>SYNOPSIS</b>    | <pre>cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [ library... ]  c89 [ flag... ] file... -lcurses [ library... ]  #include &lt;curses.h&gt;  int insnstr(const char *str, int n); int insstr(const char *str); int mvinsnstr(int y, int x, const char *str, int n); int mvinsstr(int y, int x, const char *str); int mvwinsnstr(WINDOW *win, int y, int x, const char *str, int n); int mvwinsstr(WINDOW *win, int y, int x, const char *str); int winsnstr(WINDOW *win, const char *str, int n); int winsstr(WINDOW *win, const char *str);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |            |                                            |          |                                                                                                                                  |          |                                                                          |          |                                                                             |            |                                                                   |
| <b>PARAMETERS</b>  | <table border="0"> <tr> <td style="padding-right: 20px;"><i>str</i></td> <td>Is a pointer to the string to be inserted.</td> </tr> <tr> <td><i>n</i></td> <td>Is the number of characters not to exceed when inserting <i>str</i>. If <i>n</i> is less than 1, the entire string is inserted.</td> </tr> <tr> <td><i>y</i></td> <td>Is the <i>y</i> (row) coordinate of the starting position of the string.</td> </tr> <tr> <td><i>x</i></td> <td>Is the <i>x</i> (column) coordinate of the starting position of the string.</td> </tr> <tr> <td><i>win</i></td> <td>Is a pointer to the window in which the string is to be inserted.</td> </tr> </table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | <i>str</i> | Is a pointer to the string to be inserted. | <i>n</i> | Is the number of characters not to exceed when inserting <i>str</i> . If <i>n</i> is less than 1, the entire string is inserted. | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string. | <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string. | <i>win</i> | Is a pointer to the window in which the string is to be inserted. |
| <i>str</i>         | Is a pointer to the string to be inserted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |            |                                            |          |                                                                                                                                  |          |                                                                          |          |                                                                             |            |                                                                   |
| <i>n</i>           | Is the number of characters not to exceed when inserting <i>str</i> . If <i>n</i> is less than 1, the entire string is inserted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |            |                                            |          |                                                                                                                                  |          |                                                                          |          |                                                                             |            |                                                                   |
| <i>y</i>           | Is the <i>y</i> (row) coordinate of the starting position of the string.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |            |                                            |          |                                                                                                                                  |          |                                                                          |          |                                                                             |            |                                                                   |
| <i>x</i>           | Is the <i>x</i> (column) coordinate of the starting position of the string.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |            |                                            |          |                                                                                                                                  |          |                                                                          |          |                                                                             |            |                                                                   |
| <i>win</i>         | Is a pointer to the window in which the string is to be inserted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |            |                                            |          |                                                                                                                                  |          |                                                                          |          |                                                                             |            |                                                                   |
| <b>DESCRIPTION</b> | <p>The <code>insstr()</code> function inserts <i>str</i> at the current cursor position of the <code>stdscr</code> window. The <code>winsstr()</code> function performs the identical action, but in window <i>win</i>. The <code>mvinsstr()</code> and <code>mvwinsstr()</code> functions insert the character string at the starting position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former to the <code>stdscr</code> window; the latter to window <i>win</i>).</p> <p>The <code>insnstr()</code>, <code>winsnstr()</code>, <code>mvinsnstr()</code>, and <code>mvwinsnstr()</code> functions insert <i>n</i> characters to the window or as many as will fit on the line. If <i>n</i> is less than 1, the entire string is inserted or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <i>x</i> and <i>y</i> parameters.</p> <p>All characters to the right of inserted characters are moved to the right. Characters that don't fit on the current line are discarded. The cursor is left at the point of insertion.</p> |            |                                            |          |                                                                                                                                  |          |                                                                          |          |                                                                             |            |                                                                   |

winsstr(3XCURSES)

If a character in *str* is a newline, carriage return, backspace, or tab, the cursor is moved appropriately. The cursor is moved to the next tab stop for each tab character (by default, tabs are eight characters apart). If the character is a control character other than those previously mentioned, the character is inserted using  $\^x$  notation, where *x* is a printable character. `clrtoeol(3XCURSES)` is automatically done before a newline.

**RETURN VALUES** On success, these functions return OK. Otherwise, they return ERR.

**ERRORS** None.

**ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard        |
| MT-Level            | Unsafe          |

**SEE ALSO** `addchstr(3XCURSES)`, `addstr(3XCURSES)`, `clrtoeol(3XCURSES)`, `ins_nwstr(3XCURSES)`, `insch(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| <b>NAME</b>          | curs_instr, instr, innstr, winstr, winnstr, mvinstr, mvinnstr, mvwinstr, mvwinnstr – get a string of characters from a curses window                                                                                                                                                                                                                                                                                            |                |                 |          |        |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|----------|--------|
| <b>SYNOPSIS</b>      | <pre>cc [ flag ... ] file ... -lcurses [ library ... ] #include &lt;curses.h&gt;  int instr(char *str); int innstr(char *str, int n); int winstr(WINDOW *win, char *str); int winnstr(WINDOW *win, char *str, int n); int mvinstr(int y, int x, char *str); int mvinnstr(int y, int x, char *str, int n); int mvwinstr(WINDOW *win, int y, int x, char *str); int mvwinnstr(WINDOW *win, int y, int x, char *str, int n);</pre> |                |                 |          |        |
| <b>DESCRIPTION</b>   | These routines return a string of characters in <i>str</i> , starting at the current cursor position in the named window and ending at the right margin of the window. Attributes are stripped from the characters. The four functions with <i>n</i> as the last argument return the string at most <i>n</i> characters long.                                                                                                   |                |                 |          |        |
| <b>RETURN VALUES</b> | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.                                                                                                                                                                                                                                                                                                                |                |                 |          |        |
| <b>ATTRIBUTES</b>    | See attributes(5) for descriptions of the following attributes:                                                                                                                                                                                                                                                                                                                                                                 |                |                 |          |        |
|                      | <table border="1"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table>                                                                                                                                                                                                  | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE       | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                                                                                                                 |                |                 |          |        |
| MT-Level             | Unsafe                                                                                                                                                                                                                                                                                                                                                                                                                          |                |                 |          |        |
| <b>SEE ALSO</b>      | curses(3CURSES), attributes(5)                                                                                                                                                                                                                                                                                                                                                                                                  |                |                 |          |        |
| <b>NOTES</b>         | <p>The header &lt;curses.h&gt; automatically includes the headers &lt;stdio.h&gt; and &lt;unctrl.h&gt;.</p> <p>Note that all routines except winnstr() may be macros.</p>                                                                                                                                                                                                                                                       |                |                 |          |        |

## winstr(3XCURSES)

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |            |                                                                                   |          |                                                                        |          |                                                                                          |          |                                                                                             |            |                                                                    |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|-----------------------------------------------------------------------------------|----------|------------------------------------------------------------------------|----------|------------------------------------------------------------------------------------------|----------|---------------------------------------------------------------------------------------------|------------|--------------------------------------------------------------------|
| <b>NAME</b>        | <code>innstr</code> , <code>instr</code> , <code>mvinnstr</code> , <code>mvinstr</code> , <code>mvwinnstr</code> , <code>mvwinstr</code> , <code>winnstr</code> , <code>winstr</code> – retrieve a multibyte character string (without rendition)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |            |                                                                                   |          |                                                                        |          |                                                                                          |          |                                                                                             |            |                                                                    |
| <b>SYNOPSIS</b>    | <pre>cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [ library... ]  c89 [ flag... ] file... -lcurses [ library... ]  #include &lt;curses.h&gt;  int innstr(char *str, int n); int instr(char *str); int mvinnstr(int y, int x, char *str, int n); int mvinstr(int y, int x, char *str); int mvwinnstr(WINDOW *win, int y, int x, char *str, int n); int mvwinstr(WINDOW *win, int y, int x, char *str); int winstr(WINDOW *win, char *str); int winnstr(WINDOW *win, char *str, int n);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |            |                                                                                   |          |                                                                        |          |                                                                                          |          |                                                                                             |            |                                                                    |
| <b>PARAMETERS</b>  | <table><tr><td><i>str</i></td><td>Is a pointer to an object that can hold the retrieved multibyte character string.</td></tr><tr><td><i>n</i></td><td>Is the number of characters not to exceed when retrieving <i>str</i>.</td></tr><tr><td><i>y</i></td><td>Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved.</td></tr><tr><td><i>x</i></td><td>Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved.</td></tr><tr><td><i>win</i></td><td>Is a pointer to the window in which the string is to be retrieved.</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                                                | <i>str</i> | Is a pointer to an object that can hold the retrieved multibyte character string. | <i>n</i> | Is the number of characters not to exceed when retrieving <i>str</i> . | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | <i>win</i> | Is a pointer to the window in which the string is to be retrieved. |
| <i>str</i>         | Is a pointer to an object that can hold the retrieved multibyte character string.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |            |                                                                                   |          |                                                                        |          |                                                                                          |          |                                                                                             |            |                                                                    |
| <i>n</i>           | Is the number of characters not to exceed when retrieving <i>str</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |            |                                                                                   |          |                                                                        |          |                                                                                          |          |                                                                                             |            |                                                                    |
| <i>y</i>           | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |            |                                                                                   |          |                                                                        |          |                                                                                          |          |                                                                                             |            |                                                                    |
| <i>x</i>           | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |            |                                                                                   |          |                                                                        |          |                                                                                          |          |                                                                                             |            |                                                                    |
| <i>win</i>         | Is a pointer to the window in which the string is to be retrieved.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |            |                                                                                   |          |                                                                        |          |                                                                                          |          |                                                                                             |            |                                                                    |
| <b>DESCRIPTION</b> | <p>The <code>instr()</code> and <code>winstr()</code> functions retrieve a multibyte character string (without attributes) starting at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively, and ending at the right margin. The <code>mvinstr()</code> and <code>mvwinstr()</code> functions retrieve a multibyte character string located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>innstr()</code>, <code>winnstr()</code>, <code>mvinnstr()</code>, and <code>mvwinnstr()</code> functions retrieve at most <i>n</i> characters from the window <code>stdscr</code> and <i>win</i>, respectively. The former two functions retrieve the string starting at the current cursor position; the latter two commands return the string, starting at the position specified by the <i>x</i> and <i>y</i> parameters.</p> |            |                                                                                   |          |                                                                        |          |                                                                                          |          |                                                                                             |            |                                                                    |

`winstr(3XCURSES)`

All these functions store the retrieved string in the object pointed to by *str*. They only store complete multibyte characters. If the area pointed to by *str* is not large enough to hold at least one character, these functions fail.

Only the character portion of the character/rendition pair is returned. To return the complete character/rendition pair, use `winchstr()`.

**ERRORS** OK Successful completion.

ERR An error occurred.

**USAGE** All functions except `winnstr()` may be macros.

**ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard        |
| MT-Level            | Unsafe          |

**SEE ALSO** `inch(3XCURSES)`, `inchstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

## winswch(3CURSES)

| <b>NAME</b>         | <code> curs_inswch, inswch, winswch, mvinswch, mvwinswch </code> – insert a <code>wchar_t</code> character before the character under the cursor in a curses window                                                                                                                                                                                                                                                                                 |                |                 |          |        |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|----------|--------|
| <b>SYNOPSIS</b>     | <pre>cc [ flag ... ] file ... -lcurses [ library .. ] #include &lt;curses.h&gt;  int <b>inswch</b>(chtype <i>wch</i>); int <b>winswch</b>(WINDOW *<i>win</i>, chtype <i>wch</i>); int <b>mvinswch</b>(int <i>y</i>, int <i>x</i>, chtype <i>wch</i>); int <b>mvwinswch</b>(WINDOW *<i>win</i>, int <i>y</i>, int <i>x</i>, chtype <i>wch</i>);</pre>                                                                                                |                |                 |          |        |
| <b>DESCRIPTION</b>  | These routines insert the character <i>wch</i> , holding a <code>wchar_t</code> character, before the character under the cursor. All characters to the right of the cursor are moved one space to the right, with the possibility of the rightmost character on the line being lost. The cursor position does not change (after moving to <i>y</i> , <i>x</i> , if specified). (This does not imply use of the hardware insert character feature.) |                |                 |          |        |
| <b>RETURN VALUE</b> | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion.                                                                                                                                                                                                                                                                                                          |                |                 |          |        |
| <b>ATTRIBUTES</b>   | See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1" data-bbox="444 999 1414 1089"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table>                                                                                                                                                                                | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE      | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                                                                                                                                     |                |                 |          |        |
| MT-Level            | Unsafe                                                                                                                                                                                                                                                                                                                                                                                                                                              |                |                 |          |        |
| <b>SEE ALSO</b>     | <code>curses(3CURSES)</code> , <code>attributes(5)</code>                                                                                                                                                                                                                                                                                                                                                                                           |                |                 |          |        |
| <b>NOTES</b>        | The header file <code>&lt;curses.h&gt;</code> automatically includes the header files <code>&lt;stdio.h&gt;</code> , <code>&lt;unctrl.h&gt;</code> and <code>&lt;wider.h&gt;</code> .<br><br>Note that <code>inswch()</code> , <code>mvinswch()</code> , and <code>mvwinswch()</code> may be macros.<br><br>None of these routines can use the color attribute in <code>chtype</code> .                                                             |                |                 |          |        |

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>NAME</b>          | ins_wch, wins_wch, mvins_wch, mvwins_wch – insert a complex character                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>SYNOPSIS</b>      | <pre>cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [ library... ]  c89 [ flag... ] file... -lcurses [ library... ]  #include &lt;curses.h&gt;  int ins_wch(const cchar_t *wch);  int mvins_wch(int y, int x, const cchar_t *wch);  int mvwins_wch(WINDOW *win, int y, int x, const cchar_t *wch);  int wins_wch(WINDOW *win, const cchar_t *wch);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <b>PARAMETERS</b>    | <p><i>wch</i> Is the complex character to be inserted.</p> <p><i>y</i> Is the y (row) coordinate of the position of the character.</p> <p><i>x</i> Is the x (column) coordinate of the position of the character.</p> <p><i>win</i> Is a pointer to the window in which the character is to be inserted.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>DESCRIPTION</b>   | <p>The <code>ins_wch()</code> function inserts the complex character <i>wch</i> at the current cursor position of the <code>stdscr</code> window. The <code>wins_wch()</code> function performs the identical action but in window <i>win</i>. The <code>mvins_wch()</code> and <code>mvwins_wch()</code> functions insert the character at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>). The cursor position does not change.</p> <p>All characters to the right of the inserted character are moved right one character. The last character on the line is deleted.</p> <p>Insertions and deletions occur at the character level. The cursor is adjusted to the first column of the character prior to the the operation.</p> |
| <b>RETURN VALUES</b> | On success, these functions return OK. Otherwise, they return ERR.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <b>ERRORS</b>        | None.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>ATTRIBUTES</b>    | See <code>attributes(5)</code> for descriptions of the following attributes:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard        |
| MT-Level            | Unsafe          |

wins\_wch(3XCURSES)

**SEE ALSO** | add\_wch(3XCURSES), ins\_nwstr(3XCURSES), libcurses(3XCURSES),  
attributes(5), standards(5)

| <b>NAME</b>         | curs_inswstr, inswstr, insnwstr, winswstr, winsnwstr, mvinswstr, mvinsnwstr, mvwinswstr, mvwinsnwstr – insert wchar_t string before character under the cursor in a curses window                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                |                 |          |        |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|----------|--------|
| <b>SYNOPSIS</b>     | <pre>cc [ flag ... ] file ... -lcurses [ library .. ] #include &lt;curses.h&gt;  int inswstr(wchar_t *wstr); int insnwstr(wchar_t *wstr, int n); int winswstr(WINDOW *win, wchar_t *wstr); int winsnwstr(WINDOW *win, wchar_t *wstr, int n); int mvinswstr(int y, int x, wchar_t *wstr); int mvinsnwstr(int y, int x, wchar_t *wstr, int n); int mvwinswstr(WINDOW *win, int y, int x, wchar_t *wstr); int mvwinsnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                |                 |          |        |
| <b>DESCRIPTION</b>  | <p>These routines insert a wchar_t character string (as many wchar_t characters as will fit on the line) before the character under the cursor. All characters to the right of the cursor are moved to the right, with the possibility of the rightmost characters on the line being lost. The cursor position does not change (after moving to <i>y, x</i>, if specified). (This does not imply use of the hardware insert character feature.) The four routines with <i>n</i> as the last argument insert at most <i>n</i> wchar_t characters. If <i>n</i> ≤ 0, then the entire string is inserted.</p> <p>If a character in <i>wstr</i> is a tab, newline, carriage return, or backspace, the cursor is moved appropriately within the window. A newline also does a clrtoeol(3CURSES) before moving. Tabs are considered to be at every eighth column. If a character in <i>wstr</i> is another control character, it is drawn in the ^X notation. Calling winwch(3CURSES) after adding a control character (and moving to it, if necessary) does not return the control character, but instead returns the representation of the control character.</p> |                |                 |          |        |
| <b>RETURN VALUE</b> | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                |                 |          |        |
| <b>ATTRIBUTES</b>   | See attributes(5) for descriptions of the following attributes:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                |                 |          |        |
|                     | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE      | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                |                 |          |        |
| MT-Level            | Unsafe                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                |                 |          |        |
| <b>SEE ALSO</b>     | clrtoeol(3CURSES), curses(3CURSES), winwch(3CURSES), attributes(5)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                |                 |          |        |
| <b>NOTES</b>        | The header file <curses.h> automatically includes the header files <stdio.h>, <unctrl.h> and <wchar.h>.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                |                 |          |        |

winswstr(3CURSES)

Note that all but `winswstr()` may be macros.

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |             |                                            |          |                                                                                                                                   |          |                                                                   |          |                                                                      |            |                                                                   |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|--------------------------------------------|----------|-----------------------------------------------------------------------------------------------------------------------------------|----------|-------------------------------------------------------------------|----------|----------------------------------------------------------------------|------------|-------------------------------------------------------------------|
| <b>NAME</b>        | ins_nwstr, ins_wstr, mvins_nwstr, mvins_wstr, mvwins_nwstr, mvwins_wstr, wins_nwstr, wins_wstr – insert a wide character string                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |             |                                            |          |                                                                                                                                   |          |                                                                   |          |                                                                      |            |                                                                   |
| <b>SYNOPSIS</b>    | <pre>cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [ library... ]  c89 [ flag... ] file... -lcurses [ library... ]  #include &lt;curses.h&gt;  int ins_nwstr(const wchar_t *wstr, int n); int ins_wstr(const wchar_t *wstr); int mvins_nwstr(int y, int x, const wchar_t *wstr, int n); int mvins_wstr(int y, int x, const wchar_t *wstr); int mvwins_nwstr(WINDOW *win, int y, int x, const wchar_t *wstr, int n); int mvwins_wstr(WINDOW *win, int y, int x, const wchar_t *wstr); int wins_nwstr(WINDOW *win, const wchar_t *wstr, int n); int wins_wstr(WINDOW *win, const wchar_t *wstr);</pre>                                                                                                                                                                                                                                                                                                                                                       |             |                                            |          |                                                                                                                                   |          |                                                                   |          |                                                                      |            |                                                                   |
| <b>PARAMETERS</b>  | <table border="0"> <tr> <td style="padding-right: 20px;"><i>wstr</i></td> <td>Is a pointer to the string to be inserted.</td> </tr> <tr> <td><i>n</i></td> <td>Is the number of characters not to exceed when inserting <i>wstr</i>. If <i>n</i> is less than 1, the entire string is inserted.</td> </tr> <tr> <td><i>y</i></td> <td>Is the y (row) coordinate of the starting position of the string.</td> </tr> <tr> <td><i>x</i></td> <td>Is the x (column) coordinate of the starting position of the string.</td> </tr> <tr> <td><i>win</i></td> <td>Is a pointer to the window in which the string is to be inserted.</td> </tr> </table>                                                                                                                                                                                                                                                                                                                                                      | <i>wstr</i> | Is a pointer to the string to be inserted. | <i>n</i> | Is the number of characters not to exceed when inserting <i>wstr</i> . If <i>n</i> is less than 1, the entire string is inserted. | <i>y</i> | Is the y (row) coordinate of the starting position of the string. | <i>x</i> | Is the x (column) coordinate of the starting position of the string. | <i>win</i> | Is a pointer to the window in which the string is to be inserted. |
| <i>wstr</i>        | Is a pointer to the string to be inserted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |             |                                            |          |                                                                                                                                   |          |                                                                   |          |                                                                      |            |                                                                   |
| <i>n</i>           | Is the number of characters not to exceed when inserting <i>wstr</i> . If <i>n</i> is less than 1, the entire string is inserted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |             |                                            |          |                                                                                                                                   |          |                                                                   |          |                                                                      |            |                                                                   |
| <i>y</i>           | Is the y (row) coordinate of the starting position of the string.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |             |                                            |          |                                                                                                                                   |          |                                                                   |          |                                                                      |            |                                                                   |
| <i>x</i>           | Is the x (column) coordinate of the starting position of the string.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |             |                                            |          |                                                                                                                                   |          |                                                                   |          |                                                                      |            |                                                                   |
| <i>win</i>         | Is a pointer to the window in which the string is to be inserted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |             |                                            |          |                                                                                                                                   |          |                                                                   |          |                                                                      |            |                                                                   |
| <b>DESCRIPTION</b> | <p>The <code>ins_wstr()</code> function inserts <i>wstr</i> at the current cursor position of the <code>stdscr</code> window. The <code>wins_wstr()</code> function performs the identical action, but in window <i>win</i>. The <code>mvins_wstr()</code> and <code>mvwins_wstr()</code> functions insert <i>wstr</i> string at the starting position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>ins_nwstr()</code>, <code>wins_nwstr()</code>, <code>mvins_nwstr()</code>, and <code>mvwins_nwstr()</code> functions insert <i>n</i> characters to the window or as many as will fit on the line. If <i>n</i> is less than 1, the entire string is inserted or as much of it as fits on the line. The former two functions place the string at the current cursor position; the latter two commands use the position specified by the <i>x</i> and <i>y</i> parameters.</p> |             |                                            |          |                                                                                                                                   |          |                                                                   |          |                                                                      |            |                                                                   |

wins\_wstr(3XCURSES)

All characters to the right of inserted characters are moved to the right. Characters that don't fit on the current line are discarded. The cursor is left at the point of insertion.

If a character in *wstr* is a newline, carriage return, backspace, or tab, the cursor is moved appropriately. The cursor is moved to the next tab stop for each tab character (by default, tabs are eight characters apart). If the character is a control character other than those previously mentioned, the character is inserted using  $\^x$  notation, where  $x$  is a printable character. `clrtoeol(3XCURSES)` is automatically done before a newline.

**RETURN VALUES** On success, these functions return `OK`. Otherwise, they return `ERR`.

**ERRORS** None.

**ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard        |
| MT-Level            | Unsafe          |

**SEE ALSO** `add_wchnstr(3XCURSES)`, `addnwstr(3XCURSES)`, `clrtoeol(3XCURSES)`, `ins_wch(3XCURSES)`, `insnstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| <b>NAME</b>        | curs_inwch, inwch, winwch, mvwinch, mvwinwch – get a wchar_t character and its attributes from a curses window                                                                                                                                                                                                                    |                |                 |          |        |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|----------|--------|
| <b>SYNOPSIS</b>    | <pre>cc [ flag ... ] file ... -lcurses [ library .. ] #include &lt;curses.h&gt;  ctype <b>inwch</b>(void); ctype <b>winwch</b>(WINDOW *win); ctype <b>mvwinch</b>(int y, int x); ctype <b>mvwinwch</b>(WINDOW *win, int y, int x);</pre>                                                                                          |                |                 |          |        |
| <b>DESCRIPTION</b> | These routines return the wchar_t character, of type ctype, at the current position in the named window. If any attributes are set for that position, their values are OR-ed into the value returned. Constants defined in <curses.h> can be used with the logical AND (&) operator to extract the character or attributes alone. |                |                 |          |        |
| <b>Attributes</b>  | <p>The following bit-masks may be AND-ed with characters returned by winwch().</p> <pre>A_WCHARTEXT      Bit-mask to extract character A_WATTRIBUTES    Bit-mask to extract attributes</pre>                                                                                                                                      |                |                 |          |        |
| <b>ATTRIBUTES</b>  | See attributes(5) for descriptions of the following attributes:                                                                                                                                                                                                                                                                   |                |                 |          |        |
|                    | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table>                                                    | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE     | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                   |                |                 |          |        |
| MT-Level           | Unsafe                                                                                                                                                                                                                                                                                                                            |                |                 |          |        |
| <b>SEE ALSO</b>    | curses(3CURSES), attributes(5)                                                                                                                                                                                                                                                                                                    |                |                 |          |        |
| <b>NOTES</b>       | <p>The header file &lt;curses.h&gt; automatically includes the header files &lt;stdio.h&gt;, &lt;unctrl.h&gt; and &lt;widec.h&gt;.</p> <p>Note that all of these routines may be macros.</p> <p>None of these routines can use the color attribute in ctype.</p>                                                                  |                |                 |          |        |

win\_wch(3XCURSES)

**NAME** | in\_wch, mvin\_wch, mvwin\_wch, win\_wch – retrieve a complex character (with rendition)

**SYNOPSIS** | `cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \`  
`-R /usr/xpg4/lib -lcurses [ library... ]`

`c89 [ flag... ] file... -lcurses [ library... ]`

`#include <curses.h>`

`int in_wch(cchar_t *wcv);`

`int mvin_wch(int y, int x, cchar_t *wcv);`

`int mvwin_wch(WINDOW *win, int y, cchar_t *wcv);`

`int win_wch(WINDOW *win, cchar_t *wcv);`

**DESCRIPTION** | The in\_wch() and win\_wch() functions retrieve the complex character and its rendition located at the current cursor position of the stdscr window and window win, respectively. The mvin\_wch() and mvwin\_wch() functions retrieve the complex character and its rendition located at the position indicated by the x (column) and y (row) parameters (the former in the stdscr window; the latter in window win).

All these functions store the retrieved character and its rendition in the object pointed to by wcv.

**PARAMETERS** | *wcv* | Is a pointer to an object that can store a complex character and its rendition.

| *y* | Is the y (row) coordinate of the position of the character to be returned.

| *x* | Is the x (column) coordinate of the position of the character to be returned.

| *win* | Is a pointer to the window that contains the character to be returned.

**RETURN VALUES** | On success, these functions return OK. Otherwise, they return ERR.

**ERRORS** | None.

**ATTRIBUTES** | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard        |
| MT-Level            | Unsafe          |

**SEE ALSO** | add\_wch(3XCURSES), inch(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

| <b>NAME</b>         | curs_inwchstr, inwchstr, inwchnstr, winwchstr, winwchnstr, mvinwchstr, mvinwchnstr, mvwinwchstr, mvwinwchnstr – get a string of wchar_t characters (and attributes) from a curses window                                                                                                                                                                                                                                                                                                                        |                |                 |          |        |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|----------|--------|
| <b>SYNOPSIS</b>     | <pre>cc [ flag ... ] file ... -lcurses [ library .. ] #include &lt;curses.h&gt;  int inwchstr(chtype *wchstr); int inwchnstr(chtype *wchstr, int n); int winwchstr(WINDOW *win, chtype *wchstr); int winwchnstr(WINDOW *win, chtype *wchstr, int n); int mvinwchstr(int y, int x, chtype *wchstr); int mvinwchnstr(int y, int x, chtype *wchstr, int n); int mvwinwchstr(WINDOW *win, int y, int x, chtype *wchstr); int mvwinwchnstr(WINDOW *win, int y, int x, chtype *wchstr, int n);</pre>                  |                |                 |          |        |
| <b>DESCRIPTION</b>  | These routines return a string of type chtype, holding wchar_t characters, starting at the current cursor position in the named window and ending at the right margin of the window. The four functions with <i>n</i> as the last argument, return the string at most <i>n</i> wchar_t characters long. Constants defined in <curses.h> can be used with the logical AND (&) operator to extract the wchar_t character or the attribute alone from any position in the <i>wchstr</i> (see curs_inwch(3CURSES)). |                |                 |          |        |
| <b>RETURN VALUE</b> | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.                                                                                                                                                                                                                                                                                                                                                                                                |                |                 |          |        |
| <b>ATTRIBUTES</b>   | See attributes(5) for a description of the following attributes:                                                                                                                                                                                                                                                                                                                                                                                                                                                |                |                 |          |        |
|                     | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table>                                                                                                                                                                                                                                                                                                                                          | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE      | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                |                 |          |        |
| MT-Level            | Unsafe                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                |                 |          |        |
| <b>SEE ALSO</b>     | curses(3CURSES), curs_inwch(3CURSES), attributes(5)                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                |                 |          |        |
| <b>NOTES</b>        | <p>The header file &lt;curses.h&gt; automatically includes the header files &lt;stdio.h&gt;, &lt;unctrl.h&gt; and &lt;widec.h&gt;.</p> <p>Note that all routines except winwchnstr() may be macros.</p> <p>None of these routines can use the color attribute in chtype.</p>                                                                                                                                                                                                                                    |                |                 |          |        |

## win\_wchnstr(3XCURSES)

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |               |                                                                                       |          |                                                                           |          |                                                                                          |          |                                                                                             |            |                                                                    |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|---------------------------------------------------------------------------------------|----------|---------------------------------------------------------------------------|----------|------------------------------------------------------------------------------------------|----------|---------------------------------------------------------------------------------------------|------------|--------------------------------------------------------------------|
| <b>NAME</b>        | in_wchnstr, in_wchstr, mvin_wchnstr, mvin_wchstr, mvwin_wchnstr, mvwin_wchstr, win_wchnstr, win_wchstr – retrieve complex character string (with rendition)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |               |                                                                                       |          |                                                                           |          |                                                                                          |          |                                                                                             |            |                                                                    |
| <b>SYNOPSIS</b>    | <pre>cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [ library... ]  c89 [ flag... ] file... -lcurses [ library... ]  #include &lt;curses.h&gt;  int in_wchnstr(cchar_t *wchstr, int n); int in_wchstr(cchar_t *wchstr); int mvin_wchnstr(int y, int x, cchar_t *wchstr, int n); int mvin_wchstr(int y, int x, cchar_t *wchstr); int mvwin_wchnstr(WINDOW *win, int y, int x, cchar_t *wchstr, int n); int mvwin_wchstr(WINDOW *win, int y, int x, cchar_t *wchstr); int win_wchnstr(WINDOW *win, cchar_t *wchstr, int n); int win_wchstr(WINDOW *win, cchar_t *wchstr);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |               |                                                                                       |          |                                                                           |          |                                                                                          |          |                                                                                             |            |                                                                    |
| <b>DESCRIPTION</b> | <p>The <code>in_wchstr()</code> and <code>win_wchstr()</code> functions retrieve a complex character string (with rendition) starting at the current cursor position of the <code>stdscr</code> window and window <code>win</code>, respectively, and ending at the right margin. The <code>mvin_wchstr()</code> and <code>mvwin_wchstr()</code> functions retrieve a complex character string located at the position indicated by the <code>x</code> (column) and <code>y</code> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <code>win</code>).</p> <p>The <code>in_wchnstr()</code>, <code>win_wchnstr()</code>, <code>mvin_wchnstr()</code>, and <code>mvwin_wchnstr()</code> functions retrieve at most <code>n</code> characters from the window <code>stdscr</code> and <code>win</code>, respectively. The former two functions retrieve the string, starting at the current cursor position; the latter two commands retrieve the string, starting at the position specified by the <code>x</code> and <code>y</code> parameters.</p> <p>The retrieved character string (with renditions) is stored in the object pointed to by <code>wcval</code>.</p> |               |                                                                                       |          |                                                                           |          |                                                                                          |          |                                                                                             |            |                                                                    |
| <b>PARAMETERS</b>  | <table border="0"> <tr> <td style="padding-right: 20px;"><i>wchstr</i></td> <td>Is a pointer to an object where the retrieved complex character string can be stored.</td> </tr> <tr> <td><i>n</i></td> <td>Is the number of characters not to exceed when retrieving <i>wchstr</i>.</td> </tr> <tr> <td><i>y</i></td> <td>Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved.</td> </tr> <tr> <td><i>x</i></td> <td>Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved.</td> </tr> <tr> <td><i>win</i></td> <td>Is a pointer to the window in which the string is to be retrieved.</td> </tr> </table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | <i>wchstr</i> | Is a pointer to an object where the retrieved complex character string can be stored. | <i>n</i> | Is the number of characters not to exceed when retrieving <i>wchstr</i> . | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | <i>win</i> | Is a pointer to the window in which the string is to be retrieved. |
| <i>wchstr</i>      | Is a pointer to an object where the retrieved complex character string can be stored.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |               |                                                                                       |          |                                                                           |          |                                                                                          |          |                                                                                             |            |                                                                    |
| <i>n</i>           | Is the number of characters not to exceed when retrieving <i>wchstr</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |               |                                                                                       |          |                                                                           |          |                                                                                          |          |                                                                                             |            |                                                                    |
| <i>y</i>           | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |               |                                                                                       |          |                                                                           |          |                                                                                          |          |                                                                                             |            |                                                                    |
| <i>x</i>           | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |               |                                                                                       |          |                                                                           |          |                                                                                          |          |                                                                                             |            |                                                                    |
| <i>win</i>         | Is a pointer to the window in which the string is to be retrieved.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |               |                                                                                       |          |                                                                           |          |                                                                                          |          |                                                                                             |            |                                                                    |

win\_wchnstr(3XCURSES)

**RETURN VALUES** On success, these functions return OK. Otherwise, they return ERR.

**ERRORS** None.

**ATTRIBUTES** See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard        |
| MT-Level            | Unsafe          |

**SEE ALSO** in\_wch(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

## winwchstr(3CURSES)

| <b>NAME</b>         | <code> curs_inwchstr, inwchstr, inwchnstr, winwchstr, winwchnstr, mvinwchstr, mvinwchnstr, mvwinwchstr, mvwinwchnstr</code> – get a string of <code>wchar_t</code> characters (and attributes) from a curses window                                                                                                                                                                                                                                                                                                                                                                                                                        |                |                 |          |        |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|----------|--------|
| <b>SYNOPSIS</b>     | <pre>cc [ flag ... ] file ... -lcurses [ library .. ] #include &lt;curses.h&gt;  int inwchstr ( chtype *wchstr ); int inwchnstr ( chtype *wchstr, int n ); int winwchstr ( WINDOW *win, chtype *wchstr ); int winwchnstr ( WINDOW *win, chtype *wchstr, int n ); int mvinwchstr ( int y, int x, chtype *wchstr ); int mvinwchnstr ( int y, int x, chtype *wchstr, int n ); int mvwinwchstr ( WINDOW *win, int y, int x, chtype *wchstr ); int mvwinwchnstr ( WINDOW *win, int y, int x, chtype *wchstr, int n );</pre>                                                                                                                     |                |                 |          |        |
| <b>DESCRIPTION</b>  | These routines return a string of type <code>chtype</code> , holding <code>wchar_t</code> characters, starting at the current cursor position in the named window and ending at the right margin of the window. The four functions with <code>n</code> as the last argument, return the string at most <code>n</code> <code>wchar_t</code> characters long. Constants defined in <code>&lt;curses.h&gt;</code> can be used with the logical AND ( <code>&amp;</code> ) operator to extract the <code>wchar_t</code> character or the attribute alone from any position in the <code>wchstr</code> (see <code>curs_inwch(3CURSES)</code> ). |                |                 |          |        |
| <b>RETURN VALUE</b> | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                |                 |          |        |
| <b>ATTRIBUTES</b>   | See <code>attributes(5)</code> for a description of the following attributes:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                |                 |          |        |
|                     | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE      | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                |                 |          |        |
| MT-Level            | Unsafe                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                |                 |          |        |
| <b>SEE ALSO</b>     | <code>curses(3CURSES)</code> , <code>curs_inwch(3CURSES)</code> , <code>attributes(5)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                |                 |          |        |
| <b>NOTES</b>        | The header file <code>&lt;curses.h&gt;</code> automatically includes the header files <code>&lt;stdio.h&gt;</code> , <code>&lt;unctrl.h&gt;</code> and <code>&lt;widec.h&gt;</code> .<br><br>Note that all routines except <code>winwchnstr()</code> may be macros.<br><br>None of these routines can use the <code>color</code> attribute in <code>chtype</code> .                                                                                                                                                                                                                                                                        |                |                 |          |        |

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |               |                                                                                       |          |                                                                           |          |                                                                                          |          |                                                                                             |            |                                                                    |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|---------------------------------------------------------------------------------------|----------|---------------------------------------------------------------------------|----------|------------------------------------------------------------------------------------------|----------|---------------------------------------------------------------------------------------------|------------|--------------------------------------------------------------------|
| <b>NAME</b>        | in_wchnstr, in_wchstr, mvin_wchnstr, mvin_wchstr, mvwin_wchnstr, mvwin_wchstr, win_wchnstr, win_wchstr – retrieve complex character string (with rendition)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |               |                                                                                       |          |                                                                           |          |                                                                                          |          |                                                                                             |            |                                                                    |
| <b>SYNOPSIS</b>    | <pre>cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [ library... ]  c89 [ flag... ] file... -lcurses [ library... ]  #include &lt;curses.h&gt;  int in_wchnstr(cchar_t *wchstr, int n); int in_wchstr(cchar_t *wchstr); int mvin_wchnstr(int y, int x, cchar_t *wchstr, int n); int mvin_wchstr(int y, int x, cchar_t *wchstr); int mvwin_wchnstr(WINDOW *win, int y, int x, cchar_t *wchstr, int n); int mvwin_wchstr(WINDOW *win, int y, int x, cchar_t *wchstr); int win_wchnstr(WINDOW *win, cchar_t *wchstr, int n); int win_wchstr(WINDOW *win, cchar_t *wchstr);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |               |                                                                                       |          |                                                                           |          |                                                                                          |          |                                                                                             |            |                                                                    |
| <b>DESCRIPTION</b> | <p>The <code>in_wchstr()</code> and <code>win_wchstr()</code> functions retrieve a complex character string (with rendition) starting at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively, and ending at the right margin. The <code>mvin_wchstr()</code> and <code>mvwin_wchstr()</code> functions retrieve a complex character string located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>in_wchnstr()</code>, <code>win_wchnstr()</code>, <code>mvin_wchnstr()</code>, and <code>mvwin_wchnstr()</code> functions retrieve at most <i>n</i> characters from the window <code>stdscr</code> and <i>win</i>, respectively. The former two functions retrieve the string, starting at the current cursor position; the latter two commands retrieve the string, starting at the position specified by the <i>x</i> and <i>y</i> parameters.</p> <p>The retrieved character string (with renditions) is stored in the object pointed to by <i>wcval</i>.</p> |               |                                                                                       |          |                                                                           |          |                                                                                          |          |                                                                                             |            |                                                                    |
| <b>PARAMETERS</b>  | <table border="0"> <tr> <td style="padding-right: 20px;"><i>wchstr</i></td> <td>Is a pointer to an object where the retrieved complex character string can be stored.</td> </tr> <tr> <td><i>n</i></td> <td>Is the number of characters not to exceed when retrieving <i>wchstr</i>.</td> </tr> <tr> <td><i>y</i></td> <td>Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved.</td> </tr> <tr> <td><i>x</i></td> <td>Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved.</td> </tr> <tr> <td><i>win</i></td> <td>Is a pointer to the window in which the string is to be retrieved.</td> </tr> </table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                | <i>wchstr</i> | Is a pointer to an object where the retrieved complex character string can be stored. | <i>n</i> | Is the number of characters not to exceed when retrieving <i>wchstr</i> . | <i>y</i> | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved. | <i>x</i> | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved. | <i>win</i> | Is a pointer to the window in which the string is to be retrieved. |
| <i>wchstr</i>      | Is a pointer to an object where the retrieved complex character string can be stored.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |               |                                                                                       |          |                                                                           |          |                                                                                          |          |                                                                                             |            |                                                                    |
| <i>n</i>           | Is the number of characters not to exceed when retrieving <i>wchstr</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |               |                                                                                       |          |                                                                           |          |                                                                                          |          |                                                                                             |            |                                                                    |
| <i>y</i>           | Is the <i>y</i> (row) coordinate of the starting position of the string to be retrieved.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |               |                                                                                       |          |                                                                           |          |                                                                                          |          |                                                                                             |            |                                                                    |
| <i>x</i>           | Is the <i>x</i> (column) coordinate of the starting position of the string to be retrieved.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |               |                                                                                       |          |                                                                           |          |                                                                                          |          |                                                                                             |            |                                                                    |
| <i>win</i>         | Is a pointer to the window in which the string is to be retrieved.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |               |                                                                                       |          |                                                                           |          |                                                                                          |          |                                                                                             |            |                                                                    |

win\_wchstr(3XCURSES)

**RETURN VALUES** | On success, these functions return OK. Otherwise, they return ERR.

**ERRORS** | None.

**ATTRIBUTES** | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard        |
| MT-Level            | Unsafe          |

**SEE ALSO** | `in_wch(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

| <b>NAME</b>          | curs_inwstr, inwstr, innwstr, winwstr, winnwstr, mvinwstr, mvinnwstr, mvwinwstr, mvwinnwstr – get a string of wchar_t characters from a curses window                                                                                                                                                                                                                                                                                                                |                |                 |          |        |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|----------|--------|
| <b>SYNOPSIS</b>      | <pre>cc [ flag ... ] file ... -lcurses[library .. ] #include &lt;curses.h&gt;  int inwstr(wchar_t *wstr); int innwstr(wchar_t *wstr, int n); int winwstr(WINDOW *win, wchar_t *wstr); int winnwstr(WINDOW *win, wchar_t *wstr, int n); int mvinwstr(int y, int x, wchar_t *wstr); int mvinnwstr(int y, int x, wchar_t *wstr, int n); int mvwinwstr(WINDOW *win, int y, int x, wchar_t *wstr); int mvwinnwstr(WINDOW *win, int y, int x, wchar_t *wstr, int n);</pre> |                |                 |          |        |
| <b>DESCRIPTION</b>   | These routines return the string of wchar_t characters in <i>wstr</i> starting at the current cursor position in the named window and ending at the right margin of the window. Attributes are stripped from the characters. The four functions with <i>n</i> as the last argument return the string at most <i>n</i> wchar_t characters long.                                                                                                                       |                |                 |          |        |
| <b>RETURN VALUES</b> | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.                                                                                                                                                                                                                                                                                                                                                     |                |                 |          |        |
| <b>ATTRIBUTES</b>    | See attributes(5) for descriptions of the following attributes:                                                                                                                                                                                                                                                                                                                                                                                                      |                |                 |          |        |
|                      | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table>                                                                                                                                                                                                                                                                                               | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE       | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                |                 |          |        |
| MT-Level             | Unsafe                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                |                 |          |        |
| <b>SEE ALSO</b>      | curses(3CURSES), attributes(5)                                                                                                                                                                                                                                                                                                                                                                                                                                       |                |                 |          |        |
| <b>NOTES</b>         | <p>The header file &lt;curses.h&gt; automatically includes the header files &lt;stdio.h&gt;, &lt;unctrl.h&gt; and &lt;wdec.h&gt;.</p> <p>Note that all routines except winnwstr() may be macros.</p>                                                                                                                                                                                                                                                                 |                |                 |          |        |

## winwstr(3XCURSES)

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>NAME</b>        | <code>innwstr</code> , <code>inwstr</code> , <code>mvinnwstr</code> , <code>mvinwstr</code> , <code>mvwinnwstr</code> , <code>mvwinwstr</code> , <code>winnwstr</code> , <code>winwstr</code> – retrieve a wide character string (without rendition)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>SYNOPSIS</b>    | <pre>cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [ library... ]  c89 [ flag... ] file... -lcurses [ library... ]  #include &lt;curses.h&gt;  int innwstr(wchar_t *wstr, int n); int inwstr(wchar_t *wstr); int mvinnwstr(int y, int x, wchar_t *wstr, int n); int mvinwstr(int y, int x, wchar_t *wstr); int mvwinnwstr(WINDOW*win, int y, int x, wchar_t *wstr, int n); int mvwinwstr(WINDOW*win, int y, int x, wchar_t *wstr); int winwstr(WINDOW*win, wchar_t *wstr); int winnwstr(WINDOW*win, wchar_t *wstr, int n);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>PARAMETERS</b>  | <p><i>wstr</i> Is a pointer to an object that can hold the retrieved multibyte character string.</p> <p><i>n</i> Is the number of characters not to exceed when retrieving <i>wstr</i>.</p> <p><i>y</i> Is the y (row) coordinate of the starting position of the string to be retrieved.</p> <p><i>x</i> Is the x (column) coordinate of the starting position of the string to be retrieved.</p> <p><i>win</i> Is a pointer to the window in which the string is to be retrieved.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>DESCRIPTION</b> | <p>The <code>inwstr()</code> and <code>winwstr()</code> functions retrieve a wide character string (without attributes) starting at the current cursor position of the <code>stdscr</code> window and window <i>win</i>, respectively, and ending at the right margin. The <code>mvinwstr()</code> and <code>mvwinwstr()</code> functions retrieve a wide character string located at the position indicated by the <i>x</i> (column) and <i>y</i> (row) parameters (the former in the <code>stdscr</code> window; the latter in window <i>win</i>).</p> <p>The <code>innwstr()</code>, <code>winnwstr()</code>, <code>mvinnwstr()</code>, and <code>mvwinnwstr()</code> functions retrieve at most <i>n</i> characters from the window <code>stdscr</code> and <i>win</i>, respectively. The former two functions retrieve the string starting at the current cursor position; the latter two commands return the string, starting at the position specified by the <i>x</i> and <i>y</i> parameters.</p> <p>All these functions store the retrieved string in the object pointed to by <i>wstr</i>. They only store complete wide characters. If the area pointed to by <i>wstr</i> is not large enough to hold at least one character, these functions fail.</p> |

winwstr(3XCURSES)

Only the character portion of the character/rendition pair is returned. To return the complete character/rendition pair, use `win_wchstr(3XCURSES)`.

**RETURN VALUES**

On success, the `inwstr()`, `mvinwstr()`, `mvwinwstr()`, and `winwstr()` functions return OK. Otherwise, they return ERR.

On success, the `innwstr()`, `mvinnwstr()`, `mvwinnwstr()`, and `winnwstr()` functions return the number of characters read into the string. Otherwise, they return ERR.

**ERRORS**

None.

**ATTRIBUTES**

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard        |
| MT-Level            | Unsafe          |

**SEE ALSO**

`in_wch(3XCURSES)`, `in_wchnstr(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

wmove(3CURSES)

**NAME** curs\_move, move, wmove – move curses window cursor

**SYNOPSIS**

```
cc [flag ...] file ... -lcurses [library ...]
#include <curses.h>

int move(int y, int x);
int wmove(WINDOW *win, int y, int x);
```

**DESCRIPTION** With these routines, the cursor associated with the window is moved to line *y* and column *x*. This routine does not move the physical cursor of the terminal until `refresh()` is called. The position specified is relative to the upper left-hand corner of the window, which is (0,0).

**RETURN VALUES** These routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

**ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level       | Unsafe          |

**SEE ALSO** `curs_refresh(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

**NOTES** The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `move()` may be a macro.

| <b>NAME</b>          | move, wmove – move cursor in window                                                                                                                                                                                                                                                                                                                                                                                                                                        |                |                                                                        |                     |                                                                           |            |                                                                  |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|------------------------------------------------------------------------|---------------------|---------------------------------------------------------------------------|------------|------------------------------------------------------------------|
| <b>SYNOPSIS</b>      | <pre>cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [ library... ]  c89 [ flag... ] file... -lcurses [ library... ]  #include &lt;curses.h&gt;  int move(int y, int x);  int wmove(WINDOW *win, int y, int x);</pre>                                                                                                                                                                                                             |                |                                                                        |                     |                                                                           |            |                                                                  |
| <b>PARAMETERS</b>    | <table border="0"> <tr> <td style="vertical-align: top;"><i>y</i></td> <td>Is the y (row) coordinate of the position of the cursor in the window.</td> </tr> <tr> <td style="vertical-align: top;"><i>x</i></td> <td>Is the x (column) coordinate of the position of the cursor in the window.</td> </tr> <tr> <td style="vertical-align: top;"><i>win</i></td> <td>Is a pointer to the window in which the cursor is to be written.</td> </tr> </table>                   | <i>y</i>       | Is the y (row) coordinate of the position of the cursor in the window. | <i>x</i>            | Is the x (column) coordinate of the position of the cursor in the window. | <i>win</i> | Is a pointer to the window in which the cursor is to be written. |
| <i>y</i>             | Is the y (row) coordinate of the position of the cursor in the window.                                                                                                                                                                                                                                                                                                                                                                                                     |                |                                                                        |                     |                                                                           |            |                                                                  |
| <i>x</i>             | Is the x (column) coordinate of the position of the cursor in the window.                                                                                                                                                                                                                                                                                                                                                                                                  |                |                                                                        |                     |                                                                           |            |                                                                  |
| <i>win</i>           | Is a pointer to the window in which the cursor is to be written.                                                                                                                                                                                                                                                                                                                                                                                                           |                |                                                                        |                     |                                                                           |            |                                                                  |
| <b>DESCRIPTION</b>   | The <code>move()</code> function moves the logical cursor (for <code>stdscr</code> ) to the position specified by <i>y</i> (row) and <i>x</i> (column), where the upper left corner of the window is row 0, column 0. The <code>wmove()</code> function performs the same action, but moves the cursor in the window specified by <i>win</i> . The physical cursor will not move until after a call to <code>refresh(3XCURSES)</code> or <code>doupdate(3XCURSES)</code> . |                |                                                                        |                     |                                                                           |            |                                                                  |
| <b>RETURN VALUES</b> | On success, these functions return OK. Otherwise, they return ERR.                                                                                                                                                                                                                                                                                                                                                                                                         |                |                                                                        |                     |                                                                           |            |                                                                  |
| <b>ERRORS</b>        | None.                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                |                                                                        |                     |                                                                           |            |                                                                  |
| <b>ATTRIBUTES</b>    | See <code>attributes(5)</code> for descriptions of the following attributes:                                                                                                                                                                                                                                                                                                                                                                                               |                |                                                                        |                     |                                                                           |            |                                                                  |
|                      | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table>                                                                                                                                                                                                                                           | ATTRIBUTE TYPE | ATTRIBUTE VALUE                                                        | Interface Stability | Standard                                                                  | MT-Level   | Unsafe                                                           |
| ATTRIBUTE TYPE       | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                |                                                                        |                     |                                                                           |            |                                                                  |
| Interface Stability  | Standard                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                |                                                                        |                     |                                                                           |            |                                                                  |
| MT-Level             | Unsafe                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                |                                                                        |                     |                                                                           |            |                                                                  |
| <b>SEE ALSO</b>      | <code>doupdate(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code>                                                                                                                                                                                                                                                                                                                                                |                |                                                                        |                     |                                                                           |            |                                                                  |

## wmovenextch(3CURSES)

| <b>NAME</b>         | <code>curs_alecompat</code> , <code>movenextch</code> , <code>wmovenextch</code> , <code>moveprevch</code> , <code>wmoveprevch</code> , <code>adjcurspos</code> , <code>wadjcurspos</code> – these functions are added to ALE curses library for moving the cursor by character.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                |                 |          |        |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|----------|--------|
| <b>SYNOPSIS</b>     | <pre>cc [ flag ... ] file ... -lcurses [ library .. ] #include &lt;curses.h&gt;  int  movenextch(void); int  wmovenextch(WINDOW *win); int  moveprevch(void); int  wmoveprevch(WINDOW *win); int  adjcurspos(void); int  wadjcurspos(WINDOW *win);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                |                 |          |        |
| <b>DESCRIPTION</b>  | <p><code>movenextch()</code> and <code>wmovenextch()</code> move the cursor to the next character to the right. If the next character is a multicolumn character, the cursor is positioned on the first (left-most) column of that character. The new cursor position will be on the next character, even if the cursor was originally positioned on the left-most column of a multicolumn character. Note that the simple cursor increment (<code>++x</code>) does not guarantee movement to the next character, if the cursor was originally positioned on a multicolumn character. <code>getyx(3CURSES)</code> can be used to find the new position.</p> <p><code>moveprevch()</code> and <code>wmoveprevch()</code> routines are the opposite of <code>movenextch()</code> and <code>wmovenextch()</code>, moving the cursor to the left-most column of the previous character.</p> <p><code>adjcurspos()</code> and <code>wadjcurspos()</code> move the cursor to the first(left-most) column of the multicolumn character that the cursor is presently on. If the cursor is already on the first column, or if the cursor is on a single-column character, these routines will have no effect.</p> |                |                 |          |        |
| <b>RETURN VALUE</b> | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                |                 |          |        |
| <b>ATTRIBUTES</b>   | See <code>attributes(5)</code> for descriptions of the following attributes:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                |                 |          |        |
|                     | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE      | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                |                 |          |        |
| MT-Level            | Unsafe                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                |                 |          |        |
| <b>SEE ALSO</b>     | <code>curses(3CURSES)</code> , <code>getyx(3CURSES)</code> , <code>attributes(5)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                |                 |          |        |
| <b>NOTES</b>        | The header file <code>&lt;curses.h&gt;</code> automatically includes the header files <code>&lt;stdio.h&gt;</code> , <code>&lt;unctrl.h&gt;</code> and <code>&lt;widec.h&gt;</code> .<br><br>Note that <code>movenextch()</code> , <code>moveprevch()</code> , and <code>adjcurspos()</code> may be macros.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                |                 |          |        |

| <b>NAME</b>         | curs_alecompat, movenextch, wmovenextch, moveprevch, wmoveprevch, adjcurspos, wadjcurspos – these functions are added to ALE curses library for moving the cursor by character.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                |                 |          |        |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|----------|--------|
| <b>SYNOPSIS</b>     | <pre>cc [ flag ... ] file ... -lcurses [ library .. ] #include &lt;curses.h&gt;  int movenextch(void); int wmovenextch(WINDOW *win); int moveprevch(void); int wmoveprevch(WINDOW *win); int adjcurspos(void); int wadjcurspos(WINDOW *win);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                |                 |          |        |
| <b>DESCRIPTION</b>  | <p>movenextch() and wmovenextch() move the cursor to the next character to the right. If the next character is a multicolumn character, the cursor is positioned on the first (left-most) column of that character. The new cursor position will be on the next character, even if the cursor was originally positioned on the left-most column of a multicolumn character. Note that the simple cursor increment (++x) does not guarantee movement to the next character, if the cursor was originally positioned on a multicolumn character. getyx(3CURSES) can be used to find the new position.</p> <p>moveprevch() and wmoveprevch() routines are the opposite of movenextch() and wmovenextch(), moving the cursor to the left-most column of the previous character.</p> <p>adjcurspos() and wadjcurspos() move the cursor to the first(left-most) column of the multicolumn character that the cursor is presently on. If the cursor is already on the first column, or if the cursor is on a single-column character, these routines will have no effect.</p> |                |                 |          |        |
| <b>RETURN VALUE</b> | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                |                 |          |        |
| <b>ATTRIBUTES</b>   | See attributes(5) for descriptions of the following attributes:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                |                 |          |        |
|                     | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">MT-Level</td> <td style="text-align: center;">Unsafe</td> </tr> </tbody> </table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE      | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                |                 |          |        |
| MT-Level            | Unsafe                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                |                 |          |        |
| <b>SEE ALSO</b>     | curses(3CURSES), getyx(3CURSES), attributes(5)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                |                 |          |        |
| <b>NOTES</b>        | <p>The header file &lt;curses.h&gt; automatically includes the header files &lt;stdio.h&gt;, &lt;unctrl.h&gt; and &lt;widec.h&gt;.</p> <p>Note that movenextch(), moveprevch(), and adjcurspos() may be macros.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                |                 |          |        |

## wnoutrefresh(3CURSES)

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>NAME</b>        | <code>curs_refresh</code> , <code>refresh</code> , <code>wrefresh</code> , <code>wnoutrefresh</code> , <code>doupdate</code> , <code>redrawwin</code> , <code>wredrawln</code> – refresh curses windows and lines                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>SYNOPSIS</b>    | <pre>cc [ <i>flag</i> ... ] <i>file</i> ... -lcurses [ <i>library</i> ... ] #include &lt;curses.h&gt;  int <b>refresh</b>(void);  int <b>wrefresh</b>(WINDOW *<i>win</i>);  int <b>wnoutrefresh</b>(WINDOW *<i>win</i>);  int <b>doupdate</b>(void);  int <b>redrawwin</b>(WINDOW *<i>win</i>);  int <b>wredrawln</b>(WINDOW *<i>win</i>, int <i>beg_line</i>, int <i>num_lines</i>);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <b>DESCRIPTION</b> | <p>The <code>refresh()</code> and <code>wrefresh()</code> routines (or <code>wnoutrefresh()</code> and <code>doupdate()</code>) must be called to get any output on the terminal, as other routines merely manipulate data structures. The routine <code>wrefresh()</code> copies the named window to the physical terminal screen, taking into account what is already there in order to do optimizations. The <code>refresh()</code> routine is the same, using <code>stdscr</code> as the default window. Unless <code>leaveok()</code> has been enabled, the physical cursor of the terminal is left at the location of the cursor for that window.</p> <p>The <code>wnoutrefresh()</code> and <code>doupdate()</code> routines allow multiple updates with more efficiency than <code>wrefresh()</code> alone. In addition to all the window structures, <code>curses</code> keeps two data structures representing the terminal screen: a physical screen, describing what is actually on the screen, and a virtual screen, describing what the programmer wants to have on the screen.</p> <p>The routine <code>wrefresh()</code> works by first calling <code>wnoutrefresh()</code>, which copies the named window to the virtual screen, and then calling <code>doupdate()</code>, which compares the virtual screen to the physical screen and does the actual update. If the programmer wishes to output several windows at once, a series of calls to <code>wrefresh()</code> results in alternating calls to <code>wnoutrefresh()</code> and <code>doupdate()</code>, causing several bursts of output to the screen. By first calling <code>wnoutrefresh()</code> for each window, it is then possible to call <code>doupdate()</code> once, resulting in only one burst of output, with fewer total characters transmitted and less CPU time used. If the <code>win</code> argument to <code>wrefresh()</code> is the global variable <code>curscr</code>, the screen is immediately cleared and repainted from scratch.</p> <p>The <code>redrawwin()</code> routine indicates to <code>curses</code> that some screen lines are corrupted and should be thrown away before anything is written over them. These routines could be used for programs such as editors, which want a command to redraw some part of the screen or the entire screen. The routine <code>wredrawln()</code> is preferred over <code>redrawwin()</code> where a noisy communication line exists and redrawing the entire window could be subject to even more communication noise. Just redrawing several lines offers the possibility that they would show up unblemished.</p> |

wnoutrefresh(3CURSES)

**RETURN VALUES** All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

**ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level       | Unsafe          |

**SEE ALSO** `curl_ouptops(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

**NOTES** The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `refresh()` and `redrawwin()` may be macros.

## wnoutrefresh(3XCURSES)

| <b>NAME</b>          | douupdate, refresh, wnoutrefresh, wrefresh – refresh windows and lines                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                |                 |                     |          |          |        |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|---------------------|----------|----------|--------|
| <b>SYNOPSIS</b>      | <pre>cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [ library... ]  c89 [ flag... ] file... -lcurses [ library... ]  #include &lt;curses.h&gt;  int douupdate(void);  int refresh(void);  int wnoutrefresh(WINDOW *win);  int wrefresh(WINDOW *win);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                |                 |                     |          |          |        |
| <b>DESCRIPTION</b>   | <p>The <code>refresh()</code> and <code>wrefresh()</code> functions copy <code>stdscr</code> and <code>win</code>, respectively, to the terminal screen. These functions call the <code>wnoutrefresh()</code> function to copy the specified window to <code>curscr</code> and the <code>douupdate()</code> function to do the actual update. The physical cursor is mapped to the same position as the logical cursor of the last window to update <code>curscr</code> unless <code>leaveok(3XCURSES)</code> is enabled (in which case, the cursor is placed in a position that X/Open Curses finds convenient).</p> <p>When outputting several windows at once, it is often more efficient to call the <code>wnoutrefresh()</code> and <code>douupdate()</code> functions directly. A call to <code>wnoutrefresh()</code> for each window, followed by only one call to <code>douupdate()</code> to update the screen, results in one burst of output, fewer characters sent, and less CPU time used.</p> <p>If the <code>win</code> parameter to <code>wrefresh()</code> is the global variable <code>curscr</code>, the screen is immediately cleared and repainted from scratch.</p> <p>For details on how the <code>wnoutrefresh()</code> function handles overlapping windows with broad glyphs, see the <a href="#">Overlapping Windows</a> section of the <code>curses(3XCURSES)</code> reference manual page.</p> |                |                 |                     |          |          |        |
| <b>PARAMETERS</b>    | <p><code>win</code> Is a pointer to the window in which to refresh.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                |                 |                     |          |          |        |
| <b>RETURN VALUES</b> | On success, these functions return OK. Otherwise, they return ERR.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                |                 |                     |          |          |        |
| <b>ERRORS</b>        | None.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                |                 |                     |          |          |        |
| <b>ATTRIBUTES</b>    | See <code>attributes(5)</code> for descriptions of the following attributes:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                |                 |                     |          |          |        |
|                      | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Standard</td></tr><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE       | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                |                 |                     |          |          |        |
| Interface Stability  | Standard                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                |                 |                     |          |          |        |
| MT-Level             | Unsafe                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                |                 |                     |          |          |        |
| <b>SEE ALSO</b>      | <code>clearok(3XCURSES)</code> , <code>curses(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>prefresh(3XCURSES)</code> , <code>redrawwin(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                |                 |                     |          |          |        |

| <b>NAME</b>          | curs_printw, printw, wprintw, mvprintw, mvwprintw, vwprintw – print formatted output in curses windows                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                |                 |          |        |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|----------|--------|
| <b>SYNOPSIS</b>      | <pre>cc [ flag ... ] file ... -lcurses [ library ... ] #include &lt;curses.h&gt;  int <b>printw</b>(char *fmt, /* arg */ ... ); int <b>wprintw</b>(WINDOW *win, char *fmt, /* arg */ ... ); int <b>mvprintw</b>(int y, int x, char *fmt, /* arg */ ... ); int <b>mvwprintw</b>(WINDOW *win, int y, int x, char *fmt, /* arg */... ); #include &lt;varargs.h&gt; int <b>vwprintw</b>(WINDOW *win, char *fmt, /* varlist */ ... );</pre>                                                                                                                                                                                                                                    |                |                 |          |        |
| <b>DESCRIPTION</b>   | <p>The <code>printw()</code>, <code>wprintw()</code>, <code>mvprintw()</code>, and <code>mvwprintw()</code> routines are analogous to <code>printf()</code> (see <code>printf(3C)</code>). In effect, the string that would be output by <code>printf()</code> is output instead as though <code>waddstr()</code> were used on the given window.</p> <p>The <code>vwprintw()</code> routine is analogous to <code>vprintf()</code> (see <code>vprintf(3C)</code>) and performs a <code>wprintw()</code> using a variable argument list. The third argument is a <code>va_list</code>, a pointer to a list of arguments, as defined in <code>&lt;varargs.h&gt;</code>.</p> |                |                 |          |        |
| <b>RETURN VALUES</b> | All routines return the integer <code>ERR</code> upon failure and an integer value other than <code>ERR</code> upon successful completion.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                |                 |          |        |
| <b>ATTRIBUTES</b>    | See <code>attributes(5)</code> for descriptions of the following attributes:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                |                 |          |        |
|                      | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE       | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                |                 |          |        |
| MT-Level             | Unsafe                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                |                 |          |        |
| <b>SEE ALSO</b>      | <code>curses(3CURSES)</code> , <code>printf(3C)</code> , <code>vprintf(3C)</code> , <code>attributes(5)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                |                 |          |        |
| <b>NOTES</b>         | The header <code>&lt;curses.h&gt;</code> automatically includes the headers <code>&lt;stdio.h&gt;</code> and <code>&lt;unctrl.h&gt;</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                |                 |          |        |

## wprintw(3XCURSES)

| <b>NAME</b>          | mvprintw, mvwprintw, printw, wprintw – print formatted output window                                                                                                                                                                                                                                                                                                                                                 |                |                                                                             |                     |                                                                                |            |                              |            |                                                                  |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------------------------------------------------------------------|---------------------|--------------------------------------------------------------------------------|------------|------------------------------|------------|------------------------------------------------------------------|
| <b>SYNOPSIS</b>      | <pre>cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [ library... ]  c89 [ flag... ] file... -lcurses [ library... ]  #include &lt;curses.h&gt;  int mvprintw(int y, int x, char *fmt, ...); int mvwprintw(WINDOW *win, int y, int x, char *fmt, ...); int printw(char *fmt, ...); int wprintw(WINDOW *win, char *fmt, ...);</pre>                                          |                |                                                                             |                     |                                                                                |            |                              |            |                                                                  |
| <b>PARAMETERS</b>    | <table><tr><td><i>y</i></td><td>Is the y (row) coordinate position of the string's placement in the window.</td></tr><tr><td><i>x</i></td><td>Is the x (column) coordinate position of the string's placement in the window.</td></tr><tr><td><i>fmt</i></td><td>Is a printf() format string.</td></tr><tr><td><i>win</i></td><td>Is a pointer to the window in which the string is to be written.</td></tr></table> | <i>y</i>       | Is the y (row) coordinate position of the string's placement in the window. | <i>x</i>            | Is the x (column) coordinate position of the string's placement in the window. | <i>fmt</i> | Is a printf() format string. | <i>win</i> | Is a pointer to the window in which the string is to be written. |
| <i>y</i>             | Is the y (row) coordinate position of the string's placement in the window.                                                                                                                                                                                                                                                                                                                                          |                |                                                                             |                     |                                                                                |            |                              |            |                                                                  |
| <i>x</i>             | Is the x (column) coordinate position of the string's placement in the window.                                                                                                                                                                                                                                                                                                                                       |                |                                                                             |                     |                                                                                |            |                              |            |                                                                  |
| <i>fmt</i>           | Is a printf() format string.                                                                                                                                                                                                                                                                                                                                                                                         |                |                                                                             |                     |                                                                                |            |                              |            |                                                                  |
| <i>win</i>           | Is a pointer to the window in which the string is to be written.                                                                                                                                                                                                                                                                                                                                                     |                |                                                                             |                     |                                                                                |            |                              |            |                                                                  |
| <b>DESCRIPTION</b>   | The mvprintw(), mvwprintw(), printw(), and wprintw() functions are analogous to printf(3C). The effect of these functions is as though sprintf() were used to format the string, and then waddstr(3XCURSES) were used to add that multi-byte string to the current or specified window at the current or specified cursor position.                                                                                  |                |                                                                             |                     |                                                                                |            |                              |            |                                                                  |
| <b>RETURN VALUES</b> | Upon successful completion, these functions return OK. Otherwise, they return ERR.                                                                                                                                                                                                                                                                                                                                   |                |                                                                             |                     |                                                                                |            |                              |            |                                                                  |
| <b>ERRORS</b>        | No errors are defined.                                                                                                                                                                                                                                                                                                                                                                                               |                |                                                                             |                     |                                                                                |            |                              |            |                                                                  |
| <b>ATTRIBUTES</b>    | See attributes(5) for descriptions of the following attributes:                                                                                                                                                                                                                                                                                                                                                      |                |                                                                             |                     |                                                                                |            |                              |            |                                                                  |
|                      | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Standard</td></tr><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table>                                                                                                                                                                                                      | ATTRIBUTE TYPE | ATTRIBUTE VALUE                                                             | Interface Stability | Standard                                                                       | MT-Level   | Unsafe                       |            |                                                                  |
| ATTRIBUTE TYPE       | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                                                                                                      |                |                                                                             |                     |                                                                                |            |                              |            |                                                                  |
| Interface Stability  | Standard                                                                                                                                                                                                                                                                                                                                                                                                             |                |                                                                             |                     |                                                                                |            |                              |            |                                                                  |
| MT-Level             | Unsafe                                                                                                                                                                                                                                                                                                                                                                                                               |                |                                                                             |                     |                                                                                |            |                              |            |                                                                  |
| <b>SEE ALSO</b>      | addnstr(3XCURSES), libcurses(3XCURSES), printf(3C), attributes(5), standards(5)                                                                                                                                                                                                                                                                                                                                      |                |                                                                             |                     |                                                                                |            |                              |            |                                                                  |

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>NAME</b>        | curs_refresh, refresh, wrefresh, wnoutrefresh, doupdate, redrawwin, wredrawln – refresh curses windows and lines                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>SYNOPSIS</b>    | <pre>cc [ flag ... ] file ... -lcurses [ library ... ] #include &lt;curses.h&gt;  int refresh(void);  int wrefresh(WINDOW *win);  int wnoutrefresh(WINDOW *win);  int doupdate(void);  int redrawwin(WINDOW *win);  int wredrawln(WINDOW *win, int beg_line, int num_lines);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>DESCRIPTION</b> | <p>The <code>refresh()</code> and <code>wrefresh()</code> routines (or <code>wnoutrefresh()</code> and <code>doupdate()</code>) must be called to get any output on the terminal, as other routines merely manipulate data structures. The routine <code>wrefresh()</code> copies the named window to the physical terminal screen, taking into account what is already there in order to do optimizations. The <code>refresh()</code> routine is the same, using <code>stdscr</code> as the default window. Unless <code>leaveok()</code> has been enabled, the physical cursor of the terminal is left at the location of the cursor for that window.</p> <p>The <code>wnoutrefresh()</code> and <code>doupdate()</code> routines allow multiple updates with more efficiency than <code>wrefresh()</code> alone. In addition to all the window structures, <code>curses</code> keeps two data structures representing the terminal screen: a physical screen, describing what is actually on the screen, and a virtual screen, describing what the programmer wants to have on the screen.</p> <p>The routine <code>wrefresh()</code> works by first calling <code>wnoutrefresh()</code>, which copies the named window to the virtual screen, and then calling <code>doupdate()</code>, which compares the virtual screen to the physical screen and does the actual update. If the programmer wishes to output several windows at once, a series of calls to <code>wrefresh()</code> results in alternating calls to <code>wnoutrefresh()</code> and <code>doupdate()</code>, causing several bursts of output to the screen. By first calling <code>wnoutrefresh()</code> for each window, it is then possible to call <code>doupdate()</code> once, resulting in only one burst of output, with fewer total characters transmitted and less CPU time used. If the <code>win</code> argument to <code>wrefresh()</code> is the global variable <code>curscr</code>, the screen is immediately cleared and repainted from scratch.</p> <p>The <code>redrawwin()</code> routine indicates to <code>curses</code> that some screen lines are corrupted and should be thrown away before anything is written over them. These routines could be used for programs such as editors, which want a command to redraw some part of the screen or the entire screen. The routine <code>wredrawln()</code> is preferred over <code>redrawwin()</code> where a noisy communication line exists and redrawing the entire window could be subject to even more communication noise. Just redrawing several lines offers the possibility that they would show up unblemished.</p> |

wredrawln(3CURSES)

**RETURN VALUES** All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

**ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level       | Unsafe          |

**SEE ALSO**  `curs_ouptos(3CURSES)`,  `curses(3CURSES)`,  `attributes(5)`

**NOTES** The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `refresh()` and `redrawwin()` may be macros.

| <b>NAME</b>          | redrawwin, wredrawln – redraw screen or portion of screen                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                |                 |                     |          |          |        |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|---------------------|----------|----------|--------|
| <b>SYNOPSIS</b>      | <pre>cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [ library... ]  c89 [ flag... ] file... -lcurses [ library... ]  #include &lt;curses.h&gt;  int <b>redrawwin</b>(WINDOW *win);  int <b>wredrawln</b>(WINDOW *win, int beg_line, int num_lines);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                |                 |                     |          |          |        |
| <b>PARAMETERS</b>    | <p><i>win</i> Is a pointer to the window in which to redraw.</p> <p><i>beg_line</i> Is the first line to redraw.</p> <p><i>num_lines</i> Is the number of lines to redraw.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                |                 |                     |          |          |        |
| <b>DESCRIPTION</b>   | <p>The <code>redrawwin()</code> and <code>wredrawln()</code> functions force portions of a window to be redrawn to the terminal when the next refresh operation is performed.</p> <p>The <code>redrawwin()</code> function forces the entire window <i>win</i> to be redrawn, while the <code>wredrawln()</code> function forces only <i>num_lines</i> lines starting with <i>beg_line</i> to be redrawn. Normally, refresh operations use optimization methods to reduce the actual amount of the screen to redraw based on the current screen contents. These functions tell the refresh operations not to attempt any optimization when redrawing the indicated areas.</p> <p>These functions are useful when the data that exists on the screen is believed to be corrupt and for applications such as screen editors that redraw portions of the screen.</p> |                |                 |                     |          |          |        |
| <b>RETURN VALUES</b> | On success, these functions return OK. Otherwise, they return ERR.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                |                 |                     |          |          |        |
| <b>ERRORS</b>        | None.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                |                 |                     |          |          |        |
| <b>ATTRIBUTES</b>    | See <code>attributes(5)</code> for descriptions of the following attributes:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                |                 |                     |          |          |        |
|                      | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE       | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                |                 |                     |          |          |        |
| Interface Stability  | Standard                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                |                 |                     |          |          |        |
| MT-Level             | Unsafe                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                |                 |                     |          |          |        |
| <b>SEE ALSO</b>      | <code>doupdate(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                |                 |                     |          |          |        |

## wrefresh(3CURSES)

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>NAME</b>        | <code> curs_refresh, refresh, wrefresh, wnoutrefresh, doupdate, redrawwin, wredrawln </code> – refresh curses windows and lines                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>SYNOPSIS</b>    | <pre>cc [ flag ... ] file ... -lcurses [ library ... ] #include &lt;curses.h&gt;  int refresh(void);  int wrefresh(WINDOW *win);  int wnoutrefresh(WINDOW *win);  int doupdate(void);  int redrawwin(WINDOW *win);  int wredrawln(WINDOW *win, int beg_line, int num_lines);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>DESCRIPTION</b> | <p>The <code>refresh()</code> and <code>wrefresh()</code> routines (or <code>wnoutrefresh()</code> and <code>doupdate()</code>) must be called to get any output on the terminal, as other routines merely manipulate data structures. The routine <code>wrefresh()</code> copies the named window to the physical terminal screen, taking into account what is already there in order to do optimizations. The <code>refresh()</code> routine is the same, using <code>stdscr</code> as the default window. Unless <code>leaveok()</code> has been enabled, the physical cursor of the terminal is left at the location of the cursor for that window.</p> <p>The <code>wnoutrefresh()</code> and <code>doupdate()</code> routines allow multiple updates with more efficiency than <code>wrefresh()</code> alone. In addition to all the window structures, <code>curses</code> keeps two data structures representing the terminal screen: a physical screen, describing what is actually on the screen, and a virtual screen, describing what the programmer wants to have on the screen.</p> <p>The routine <code>wrefresh()</code> works by first calling <code>wnoutrefresh()</code>, which copies the named window to the virtual screen, and then calling <code>doupdate()</code>, which compares the virtual screen to the physical screen and does the actual update. If the programmer wishes to output several windows at once, a series of calls to <code>wrefresh()</code> results in alternating calls to <code>wnoutrefresh()</code> and <code>doupdate()</code>, causing several bursts of output to the screen. By first calling <code>wnoutrefresh()</code> for each window, it is then possible to call <code>doupdate()</code> once, resulting in only one burst of output, with fewer total characters transmitted and less CPU time used. If the <code>win</code> argument to <code>wrefresh()</code> is the global variable <code>curscr</code>, the screen is immediately cleared and repainted from scratch.</p> <p>The <code>redrawwin()</code> routine indicates to <code>curses</code> that some screen lines are corrupted and should be thrown away before anything is written over them. These routines could be used for programs such as editors, which want a command to redraw some part of the screen or the entire screen. The routine <code>wredrawln()</code> is preferred over <code>redrawwin()</code> where a noisy communication line exists and redrawing the entire window could be subject to even more communication noise. Just redrawing several lines offers the possibility that they would show up unblemished.</p> |

wrefresh(3CURSES)

**RETURN VALUES** All routines return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

**ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level       | Unsafe          |

**SEE ALSO** `curl_ouptops(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

**NOTES** The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `refresh()` and `redrawwin()` may be macros.

## wrefresh(3XCURSES)

| <b>NAME</b>          | douupdate, refresh, wnoutrefresh, wrefresh – refresh windows and lines                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                |                 |                     |          |          |        |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|---------------------|----------|----------|--------|
| <b>SYNOPSIS</b>      | <pre>cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [ library... ]  c89 [ flag... ] file... -lcurses [ library... ]  #include &lt;curses.h&gt;  int douupdate(void); int refresh(void); int wnoutrefresh(WINDOW *win); int wrefresh(WINDOW *win);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                |                 |                     |          |          |        |
| <b>DESCRIPTION</b>   | <p>The <code>refresh()</code> and <code>wrefresh()</code> functions copy <code>stdscr</code> and <code>win</code>, respectively, to the terminal screen. These functions call the <code>wnoutrefresh()</code> function to copy the specified window to <code>curscr</code> and the <code>douupdate()</code> function to do the actual update. The physical cursor is mapped to the same position as the logical cursor of the last window to update <code>curscr</code> unless <code>leaveok(3XCURSES)</code> is enabled (in which case, the cursor is placed in a position that X/Open Curses finds convenient).</p> <p>When outputting several windows at once, it is often more efficient to call the <code>wnoutrefresh()</code> and <code>douupdate()</code> functions directly. A call to <code>wnoutrefresh()</code> for each window, followed by only one call to <code>douupdate()</code> to update the screen, results in one burst of output, fewer characters sent, and less CPU time used.</p> <p>If the <code>win</code> parameter to <code>wrefresh()</code> is the global variable <code>curscr</code>, the screen is immediately cleared and repainted from scratch.</p> <p>For details on how the <code>wnoutrefresh()</code> function handles overlapping windows with broad glyphs, see the <a href="#">Overlapping Windows</a> section of the <code>curses(3XCURSES)</code> reference manual page.</p> |                |                 |                     |          |          |        |
| <b>PARAMETERS</b>    | <p><code>win</code> Is a pointer to the window in which to refresh.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                |                 |                     |          |          |        |
| <b>RETURN VALUES</b> | On success, these functions return OK. Otherwise, they return ERR.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                |                 |                     |          |          |        |
| <b>ERRORS</b>        | None.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                |                 |                     |          |          |        |
| <b>ATTRIBUTES</b>    | See <code>attributes(5)</code> for descriptions of the following attributes:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                |                 |                     |          |          |        |
|                      | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Standard</td></tr><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE       | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                |                 |                     |          |          |        |
| Interface Stability  | Standard                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                |                 |                     |          |          |        |
| MT-Level             | Unsafe                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                |                 |                     |          |          |        |
| <b>SEE ALSO</b>      | <code>clearok(3XCURSES)</code> , <code>curses(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>prefresh(3XCURSES)</code> , <code>redrawwin(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                |                 |                     |          |          |        |

| <b>NAME</b>          | curs_scanw, scanw, wscanw, mvscanw, mvwscanw, vwscanw – convert formatted input from a curses widow                                                                                                                                                                                                                                                                                                                                                                                                                     |                |                 |          |        |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|----------|--------|
| <b>SYNOPSIS</b>      | <pre>cc [ flag ... ] file ... -lcurses [ library ... ] #include &lt;curses.h&gt;  int scanw(char *fmt, /* arg */ ...); int wscanw(WINDOW *win, char *fmt, /* arg */ ...); int mvscanw(int y, int x, char *fmt, /* arg */ ...); int mvwscanw(WINDOW *win, int y, int x, char *fmt, /* arg */...); int vwscanw(WINDOW *win, char *fmt, va_list varlist);</pre>                                                                                                                                                            |                |                 |          |        |
| <b>DESCRIPTION</b>   | <p>The scanw(), wscanw(), and mvscanw() routines correspond to scanf() (see scanf(3C)). The effect of these routines is as though wgetstr() were called on the window, and the resulting line used as input for the scan. Fields which do not map to a variable in the fmt field are lost.</p> <p>The vwscanw() routine is similar to vwprintw() in that it performs a wscanw() using a variable argument list. The third argument is a va_list, a pointer to a list of arguments, as defined in &lt;varargs.h&gt;.</p> |                |                 |          |        |
| <b>RETURN VALUES</b> | <p>vwscanw() returns ERR on failure and an integer equal to the number of fields scanned on success.</p> <p>Applications may interrogate the return value from the scanw, wscanw(), mvscanw(), and mvwscanw() routines to determine the number of fields which were mapped in the call.</p>                                                                                                                                                                                                                             |                |                 |          |        |
| <b>ATTRIBUTES</b>    | See attributes(5) for descriptions of the following attributes:                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                |                 |          |        |
|                      | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table>                                                                                                                                                                                                                                                                                                                                                  | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE       | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                |                 |          |        |
| MT-Level             | Unsafe                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                |                 |          |        |
| <b>SEE ALSO</b>      | curs_getstr(3CURSES), curs_printw(3CURSES), curses(3CURSES), scanf(3C), attributes(5)                                                                                                                                                                                                                                                                                                                                                                                                                                   |                |                 |          |        |
| <b>NOTES</b>         | The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.                                                                                                                                                                                                                                                                                                                                                                                                                                      |                |                 |          |        |

wscanw(3XCURSES)

**NAME** mvscanw, mvwscanw, scanw, wscanw – convert formatted input from a window

**SYNOPSIS** `cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \`  
`-R /usr/xpg4/lib -lcurses [ library... ]`

**c89** `[ flag... ] file... -lcurses [ library... ]`

```
#include <curses.h>

int mvscanw(int y, int x, char *fmt, ...);
int mvwscanw(WINDOW *win, int y, int x, char *fmt, ...);
int scanw(char *fmt, ...);
int wscanw(WINDOW *win, char *fmt, ...);
```

**PARAMETERS**

*y* Is the y (row) coordinate of the position of the character to be read.

*x* Is the x (column) coordinate of the position of the character to be read.

*fmt* Is a scanf () format string.

*win* Is a pointer to the window in which the character is to be read.

**DESCRIPTION** These functions are similar to scanf(3C). Their effect is as though mvwgetstr(3XCURSES) were called to get a multi-byte character string from the current or specified window at the current or specified cursor position, and then sscanf () were used to interpret and convert that string.

**RETURN VALUES** Upon successful completion, these functions return OK. Otherwise, they return ERR.

**ERRORS** No errors are defined.

**ATTRIBUTES** See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard        |
| MT-Level            | Unsafe          |

**SEE ALSO** getnstr(3XCURSES), libcurses(3XCURSES), printw(3XCURSES), scanf(3C), wcstombs(3C), attributes(5), standards(5)

| <b>NAME</b>          | curs_scroll, scroll, scl, wscrl – scroll a curses window                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                |                 |          |        |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|----------|--------|
| <b>SYNOPSIS</b>      | <pre>cc [ flag ... ] file ... -lcurses [ library ... ] #include &lt;curses.h&gt;  int <b>scroll</b>(WINDOW *win); int <b>scl</b>(int n); int <b>wscrl</b>(WINDOW *win, int n);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                |                 |          |        |
| <b>DESCRIPTION</b>   | <p>With the <code>scroll()</code> routine, the window is scrolled up one line. This involves moving the lines in the window data structure. As an optimization, if the scrolling region of the window is the entire screen, the physical screen is scrolled at the same time.</p> <p>With the <code>scl()</code> and <code>wscrl()</code> routines, for positive <math>n</math> scroll the window up <math>n</math> lines (line <math>i+n</math> becomes <math>i</math>); otherwise scroll the window down <math>n</math> lines. This involves moving the lines in the window character image structure. The current cursor position is not changed.</p> <p>For these functions to work, scrolling must be enabled via <code>scrollok()</code>.</p> |                |                 |          |        |
| <b>RETURN VALUES</b> | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                |                 |          |        |
| <b>ATTRIBUTES</b>    | See <code>attributes(5)</code> for descriptions of the following attributes:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                |                 |          |        |
|                      | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE       | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                |                 |          |        |
| MT-Level             | Unsafe                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                |                 |          |        |
| <b>SEE ALSO</b>      | <code>curs_outopts(3CURSES)</code> , <code>curses(3CURSES)</code> , <code>attributes(5)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                |                 |          |        |
| <b>NOTES</b>         | <p>The header <code>&lt;curses.h&gt;</code> automatically includes the headers <code>&lt;stdio.h&gt;</code> and <code>&lt;unctrl.h&gt;</code>.</p> <p>Note that <code>scl()</code> and <code>scroll()</code> may be macros.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                |                 |          |        |

wscrl(3XCURSES)

**NAME** | scl, scroll, wscrl – scroll a window

**SYNOPSIS** | `cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \`  
| `-R /usr/xpg4/lib -lcurses [ library... ]`

| `c89 [ flag... ] file... -lcurses [ library... ]`

| `#include <curses.h>`  
| `int scl(int n);`  
| `int scroll(WINDOW *win);`  
| `int wscrl(WINDOW *win, int n);`

**PARAMETERS** | *n*                                    number and direction of lines to scroll  
| *win*                                    pointer to the window in which to scroll

**DESCRIPTION** | The scroll() function scrolls the window *win* up one line. The current cursor position is not changed.

| The scl() and wscrl() functions scroll the window *stdscr* or *win* up or down *n* lines, where *n* is a positive (scroll up) or negative (scroll down) integer.

| The scrollok(3XCURSES) function must be enabled for these functions to work.

**RETURN VALUES** | On success, these functions return OK. Otherwise, they return ERR.

**ERRORS** | None.

**ATTRIBUTES** | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard        |
| MT-Level            | Unsafe          |

**SEE ALSO** | clearok(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>NAME</b>        | curs_ouptoks, clearok, idlok, idcok, immedok, leaveok, setscreg, wsetscreg, scrollok, nl, nonl – curses terminal output option control routines                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>SYNOPSIS</b>    | <pre>cc [ flag ... ] file ... -lcurses [ library ... ] #include &lt;curses.h&gt;  int clearok(WINDOW *win, bool bf); int idlok(WINDOW *win, bool bf); void idcok(WINDOW *win, bool bf); void immedok(WINDOW *win, bool bf); int leaveok(WINDOW *win, bool bf); int setscreg(int top, int bot); int wsetscreg(WINDOW *win, int top, int bot); int scrollok(WINDOW *win, bool bf); int nl(void); int nonl(void);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>DESCRIPTION</b> | <p>These routines set options that deal with output within curses. All options are initially FALSE, unless otherwise stated. It is not necessary to turn these options off before calling <code>endwin()</code>.</p> <p>With the <code>clearok()</code> routine, if enabled (<i>bf</i> is TRUE), the next call to <code>wrefresh()</code> with this window will clear the screen completely and redraw the entire screen from scratch. This is useful when the contents of the screen are uncertain, or in some cases for a more pleasing visual effect. If the <i>win</i> argument to <code>clearok()</code> is the global variable <code>curscr()</code>, the next call to <code>wrefresh()</code> with any window causes the screen to be cleared and repainted from scratch.</p> <p>With the <code>idlok()</code> routine, if enabled (<i>bf</i> is TRUE), <code>curses</code> considers using the hardware insert/delete line feature of terminals so equipped. If disabled (<i>bf</i> is FALSE), <code>curses</code> very seldom uses this feature. (The insert/delete character feature is always considered.) This option should be enabled only if the application needs insert/delete line, for example, for a screen editor. It is disabled by default because insert/delete line tends to be visually annoying when used in applications where it isn't really needed. If insert/delete line cannot be used, <code>curses</code> redraws the changed portions of all lines.</p> <p>With the <code>idcok()</code> routine, if enabled (<i>bf</i> is TRUE), <code>curses</code> considers using the hardware insert/delete character feature of terminals so equipped. This is enabled by default.</p> <p>With the <code>immedok()</code> routine, if enabled (<i>bf</i> is TRUE), any change in the window image, such as the ones caused by <code>waddch()</code>, <code>wclrtoobot()</code>, <code>wscrl()</code>, etc., automatically cause a call to <code>wrefresh()</code>. However, it may degrade the performance considerably, due to repeated calls to <code>wrefresh()</code>. It is disabled by default.</p> |

## wsetscrreg(3CURSES)

Normally, the hardware cursor is left at the location of the window cursor being refreshed. The `leaveok()` option allows the cursor to be left wherever the update happens to leave it. It is useful for applications where the cursor is not used, since it reduces the need for cursor motions. If possible, the cursor is made invisible when this option is enabled.

The `setscrreg()` and `wsetscrreg()` routines allow the application programmer to set a software scrolling region in a window. *top* and *bot* are the line numbers of the top and bottom margin of the scrolling region. (Line 0 is the top line of the window.) If this option and `scrollok()` are enabled, an attempt to move off the bottom margin line causes all lines in the scrolling region to scroll up one line. Only the text of the window is scrolled. (Note that this has nothing to do with the use of a physical scrolling region capability in the terminal, like that in the VT100. If `idlok()` is enabled and the terminal has either a scrolling region or insert/delete line capability, they will probably be used by the output routines.)

The `scrollok()` option controls what happens when the cursor of a window is moved off the edge of the window or scrolling region, either as a result of a newline action on the bottom line, or typing the last character of the last line. If disabled, (*bf* is FALSE), the cursor is left on the bottom line. If enabled, (*bf* is TRUE), `wrefresh()` is called on the window, and the physical terminal and window are scrolled up one line. (Note that in order to get the physical scrolling effect on the terminal, it is also necessary to call `idlok()`.)

The `nl()` and `nonl()` routines control whether newline is translated into carriage return and linefeed on output, and whether return is translated into newline on input. Initially, the translations do occur. By disabling these translations using `nonl()`, `curses` is able to make better use of the linefeed capability, resulting in faster cursor motion.

**RETURN VALUES** `setscrreg()` and `wsetscrreg()` return OK upon success and ERR upon failure. All other routines that return an integer always return OK.

**ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level       | Unsafe          |

**SEE ALSO** `curs_addch(3CURSES)`, `curs_clear(3CURSES)`, `curs_initscr(3CURSES)`, `curs_refresh(3CURSES)`, `curs_scroll(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

**NOTES** The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `clearok()`, `leaveok()`, `scrollok()`, `idcok()`, `nl()`, `nonl()`, and `setscrreg()` may be macros.

wsetscrreg(3CURSES)

The `immedok()` routine is useful for windows that are used as terminal emulators.

## wsetscrreg(3XCURSES)

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>NAME</b>        | clearok, idlok, leaveok, scrollok, setscrreg, wsetscrreg – terminal output control functions                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>SYNOPSIS</b>    | <pre>cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [ library... ]  c89 [ flag... ] file... -lcurses [ library... ]  #include &lt;curses.h&gt;  int clearok(WINDOW *win, bool bf); int idlok(WINDOW *win, bool bf); int leaveok(WINDOW *win, bool bf); int scrollok(WINDOW *win, bool bf); int setscrreg(int top, int bot); int wsetscrreg(WINDOW *win, int top, int bot);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>DESCRIPTION</b> | <p>These functions set options that deal with the output within Curses functions.</p> <p>The <code>clearok()</code> function assigns the value of <i>bf</i> to an internal flag in the specified window that governs clearing of the screen during a refresh. If, during a refresh operation on the specified window, the flag in <i>curscr</i> is TRUE or the flag in the specified window is TRUE, <code>clearok()</code> clears the screen, redraws it in its entirety, and sets the flag to FALSE in <i>curscr</i> and in the specified window. The initial state is unspecified</p> <p>The <code>idlok()</code> function specifies whether the implementation may use the hardware insert-line, delete-line, and scroll features of terminals so equipped. If <i>bf</i> is TRUE, use of these features is enabled. If <i>bf</i> is FALSE, use of these features is disabled and lines are instead redrawn as required. The initial state is FALSE.</p> <p>The <code>leaveok()</code> function controls the cursor position after a refresh operation. If <i>bf</i> is TRUE, refresh operations on the specified window may leave the terminal's cursor at an arbitrary position. If <i>bf</i> is FALSE, then at the end of any refresh operation, the terminal's cursor is positioned at the cursor position contained in the specified window. The initial state is FALSE.</p> <p>The <code>scrollok()</code> function controls the use of scrolling. If <i>bf</i> is TRUE, then scrolling is enabled for the specified window. If <i>bf</i> is FALSE, scrolling is disabled for the specified window. The initial state is FALSE.</p> <p>The <code>setscrreg()</code> and <code>wsetscrreg()</code> functions define a software scrolling region in the current or specified window. The <i>top</i> and <i>bottom</i> arguments are the line numbers of the first and last line defining the scrolling region. (Line 0 is the top line of the window.) If this option and <code>scrollok()</code> are enabled, an attempt to move off the last line of the margin causes all lines in the scrolling region to scroll one line in the</p> |

wsetscrreg(3XCURSES)

direction of the first line. Only characters in the window are scrolled. If a software scrolling region is set and `scrollok()` is not enabled, an attempt to move off the last line of the margin does not reposition any lines in the scrolling region.

**PARAMETERS**

*win* Is a pointer to a window.  
*bf* Is a Boolean expression.  
*top* Is the top line of the scrolling region (top of the window is line 0).  
*bot* Is the bottom line of the scrolling region (top of the window is line 0).

**RETURN VALUES**

Upon successful completion, the `setscrreg()` and `wsetscrreg()` functions return OK. Otherwise, they return ERR.

The other functions always return OK.

**ERRORS**

No errors are defined.

**USAGE**

The only reason to enable the `idlok()` feature is to use scrolling to achieve the visual effect of motion of a partial window, such as for a screen editor. In other cases, the feature can be visually annoying.

The `leaveok()` option provides greater efficiency for applications that do not use the cursor.

**ATTRIBUTES**

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard        |
| MT-Level            | Unsafe          |

**SEE ALSO**

`bkgdset(3XCURSES)`, `clear(3XCURSES)`, `doupdate(3XCURSES)`, `libcurses(3XCURSES)`, `scr1(3XCURSES)`, `attributes(5)`, `standards(5)`

## wstandend(3CURSES)

|                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                         |                                        |                          |             |                        |               |                      |          |                    |             |                     |                      |                           |                         |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|----------------------------------------|--------------------------|-------------|------------------------|---------------|----------------------|----------|--------------------|-------------|---------------------|----------------------|---------------------------|-------------------------|
| <b>NAME</b>               | <code>curs_attr</code> , <code>attroff</code> , <code>wattroff</code> , <code>attron</code> , <code>wattron</code> , <code>attrset</code> , <code>wattrset</code> , <code>standend</code> , <code>wstandend</code> , <code>standout</code> , <code>wstandout</code> – curses character and window attribute control routines                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                         |                                        |                          |             |                        |               |                      |          |                    |             |                     |                      |                           |                         |
| <b>SYNOPSIS</b>           | <pre>cc [ flag ... ] file ... -lcurses [ library ... ] #include &lt;curses.h&gt;  int attroff(int attrs); int wattroff(WINDOW *win, int attrs); int attron(int attrs); int wattron(WINDOW *win, int attrs); int attrset(int attrs); int wattrset(WINDOW *win, int attrs); int standend(void); int wstandend(WINDOW *win); int standout(void); int wstandout(WINDOW *win);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                         |                                        |                          |             |                        |               |                      |          |                    |             |                     |                      |                           |                         |
| <b>DESCRIPTION</b>        | <p>All of these routines manipulate the current attributes of the named window. The current attributes of a window are applied to all characters that are written into the window with <code>waddch()</code>, <code>waddstr()</code>, and <code>wprintw()</code>. Attributes are a property of the character, and move with the character through any scrolling and insert/delete line/character operations. To the extent possible on the particular terminal, they are displayed as the graphic rendition of characters put on the screen.</p> <p>The routine <code>attrset()</code> sets the current attributes of the given window to <i>attrs</i>. The routine <code>attroff()</code> turns off the named attributes without turning any other attributes on or off. The routine <code>attron()</code> turns on the named attributes without affecting any others. The routine <code>standout()</code> is the same as <code>attron(A_STANDOUT)</code>. The routine <code>standend()</code> is the same as <code>attrset()</code>, that is, it turns off all attributes.</p> |                         |                                        |                          |             |                        |               |                      |          |                    |             |                     |                      |                           |                         |
| <b>Attributes</b>         | <p>The following video attributes, defined in <code>&lt;curses.h&gt;</code>, can be passed to the routines <code>attron()</code>, <code>attroff()</code>, and <code>attrset()</code>, or OR-ed with the characters passed to <code>addch()</code>.</p> <table><tr><td><code>A_STANDOUT</code></td><td>Best highlighting mode of the terminal</td></tr><tr><td><code>A_UNDERLINE</code></td><td>Underlining</td></tr><tr><td><code>A_REVERSE</code></td><td>Reverse video</td></tr><tr><td><code>A_BLINK</code></td><td>Blinking</td></tr><tr><td><code>A_DIM</code></td><td>Half bright</td></tr><tr><td><code>A_BOLD</code></td><td>Extra bright or bold</td></tr><tr><td><code>A_ALTCHARSET</code></td><td>Alternate character set</td></tr></table>                                                                                                                                                                                                                                                                                                                           | <code>A_STANDOUT</code> | Best highlighting mode of the terminal | <code>A_UNDERLINE</code> | Underlining | <code>A_REVERSE</code> | Reverse video | <code>A_BLINK</code> | Blinking | <code>A_DIM</code> | Half bright | <code>A_BOLD</code> | Extra bright or bold | <code>A_ALTCHARSET</code> | Alternate character set |
| <code>A_STANDOUT</code>   | Best highlighting mode of the terminal                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                         |                                        |                          |             |                        |               |                      |          |                    |             |                     |                      |                           |                         |
| <code>A_UNDERLINE</code>  | Underlining                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                         |                                        |                          |             |                        |               |                      |          |                    |             |                     |                      |                           |                         |
| <code>A_REVERSE</code>    | Reverse video                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                         |                                        |                          |             |                        |               |                      |          |                    |             |                     |                      |                           |                         |
| <code>A_BLINK</code>      | Blinking                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                         |                                        |                          |             |                        |               |                      |          |                    |             |                     |                      |                           |                         |
| <code>A_DIM</code>        | Half bright                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                         |                                        |                          |             |                        |               |                      |          |                    |             |                     |                      |                           |                         |
| <code>A_BOLD</code>       | Extra bright or bold                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                         |                                        |                          |             |                        |               |                      |          |                    |             |                     |                      |                           |                         |
| <code>A_ALTCHARSET</code> | Alternate character set                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                         |                                        |                          |             |                        |               |                      |          |                    |             |                     |                      |                           |                         |

wstandend(3CURSES)

A\_CHARTEXT Bit-mask to extract a character

COLOR\_PAIR(*n*) Color-pair number *n*

The following macro is the reverse of COLOR\_PAIR(*n*) :

PAIR\_NUMBER(*attrs*) Returns the pair number associated with the COLOR\_PAIR(*n*) attribute

**RETURN VALUES** These routines always return 1.

**ATTRIBUTES** See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level       | Unsafe          |

**SEE ALSO** curs\_addch(3CURSES), curs\_addstr(3CURSES), curs\_printw(3CURSES), curses(3CURSES), attributes(5)

**NOTES** The header < curses.h > automatically includes the headers < stdio.h > and < unctrl.h >.

Note that attroff(), wattroff(), attron(), wattron(), wattrset(), standend(), and standout() may be macros.

## wstandend(3XCURSES)

| <b>NAME</b>          | standend, standout, wstandend, wstandout – set/clear window attributes                                                                                                                                                                                                                             |                |                 |                     |          |          |        |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|---------------------|----------|----------|--------|
| <b>SYNOPSIS</b>      | <pre>cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [ library... ]  c89 [ flag... ] file... -lcurses [ library... ]  #include &lt;curses.h&gt;  int standend(void);  int standout(void);  int wstandend(WINDOW *win);  int wstandout(WINDOW *win);</pre> |                |                 |                     |          |          |        |
| <b>PARAMETERS</b>    | <i>win</i> Is a pointer to the window in which attribute changes are to be made.                                                                                                                                                                                                                   |                |                 |                     |          |          |        |
| <b>DESCRIPTION</b>   | <p>The standend() and wstandend() functions turn off all attributes associated with stdscr and <i>win</i> respectively.</p> <p>The standout() and wstandout() functions turn on the A_STANDOUT attribute of stdscr and <i>win</i> respectively.</p>                                                |                |                 |                     |          |          |        |
| <b>RETURN VALUES</b> | These functions always return 1.                                                                                                                                                                                                                                                                   |                |                 |                     |          |          |        |
| <b>ERRORS</b>        | None.                                                                                                                                                                                                                                                                                              |                |                 |                     |          |          |        |
| <b>ATTRIBUTES</b>    | See attributes(5) for descriptions of the following attributes:                                                                                                                                                                                                                                    |                |                 |                     |          |          |        |
|                      | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Standard</td></tr><tr><td>MT-Level</td><td>Unsafe</td></tr></tbody></table>                                                                                    | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE       | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                    |                |                 |                     |          |          |        |
| Interface Stability  | Standard                                                                                                                                                                                                                                                                                           |                |                 |                     |          |          |        |
| MT-Level             | Unsafe                                                                                                                                                                                                                                                                                             |                |                 |                     |          |          |        |
| <b>SEE ALSO</b>      | attr_get(3XCURSES), attroff(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)                                                                                                                                                                                                            |                |                 |                     |          |          |        |

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |            |                                        |             |             |           |               |         |          |       |             |        |                      |              |                         |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|----------------------------------------|-------------|-------------|-----------|---------------|---------|----------|-------|-------------|--------|----------------------|--------------|-------------------------|
| <b>NAME</b>        | curs_attr, attroff, wattroff, attron, wattron, attrset, wattrset, standend, wstandend, standout, wstandout – curses character and window attribute control routines                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |            |                                        |             |             |           |               |         |          |       |             |        |                      |              |                         |
| <b>SYNOPSIS</b>    | <pre>cc [ flag ... ] file ... -lcurses [ library ... ] #include &lt;curses.h&gt;  int attroff(int attrs);  int wattroff(WINDOW *win, int attrs);  int attron(int attrs);  int wattron(WINDOW *win, int attrs);  int attrset(int attrs);  int wattrset(WINDOW *win, int attrs);  int standend(void);  int wstandend(WINDOW *win);  int standout(void);  int wstandout(WINDOW *win);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |            |                                        |             |             |           |               |         |          |       |             |        |                      |              |                         |
| <b>DESCRIPTION</b> | <p>All of these routines manipulate the current attributes of the named window. The current attributes of a window are applied to all characters that are written into the window with <code>waddch()</code>, <code>waddstr()</code>, and <code>wprintw()</code>. Attributes are a property of the character, and move with the character through any scrolling and insert/delete line/character operations. To the extent possible on the particular terminal, they are displayed as the graphic rendition of characters put on the screen.</p> <p>The routine <code>attrset()</code> sets the current attributes of the given window to <code>attrs</code>. The routine <code>attroff()</code> turns off the named attributes without turning any other attributes on or off. The routine <code>attron()</code> turns on the named attributes without affecting any others. The routine <code>standout()</code> is the same as <code>attron(A_STANDOUT)</code>. The routine <code>standend()</code> is the same as <code>attrset()</code>, that is, it turns off all attributes.</p> |            |                                        |             |             |           |               |         |          |       |             |        |                      |              |                         |
| <b>Attributes</b>  | <p>The following video attributes, defined in <code>&lt;curses.h&gt;</code>, can be passed to the routines <code>attron()</code>, <code>attroff()</code>, and <code>attrset()</code>, or OR-ed with the characters passed to <code>addch()</code>.</p> <table border="0"> <tr> <td>A_STANDOUT</td> <td>Best highlighting mode of the terminal</td> </tr> <tr> <td>A_UNDERLINE</td> <td>Underlining</td> </tr> <tr> <td>A_REVERSE</td> <td>Reverse video</td> </tr> <tr> <td>A_BLINK</td> <td>Blinking</td> </tr> <tr> <td>A_DIM</td> <td>Half bright</td> </tr> <tr> <td>A_BOLD</td> <td>Extra bright or bold</td> </tr> <tr> <td>A_ALTCHARSET</td> <td>Alternate character set</td> </tr> </table>                                                                                                                                                                                                                                                                                                                                                                                    | A_STANDOUT | Best highlighting mode of the terminal | A_UNDERLINE | Underlining | A_REVERSE | Reverse video | A_BLINK | Blinking | A_DIM | Half bright | A_BOLD | Extra bright or bold | A_ALTCHARSET | Alternate character set |
| A_STANDOUT         | Best highlighting mode of the terminal                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |            |                                        |             |             |           |               |         |          |       |             |        |                      |              |                         |
| A_UNDERLINE        | Underlining                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |            |                                        |             |             |           |               |         |          |       |             |        |                      |              |                         |
| A_REVERSE          | Reverse video                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |            |                                        |             |             |           |               |         |          |       |             |        |                      |              |                         |
| A_BLINK            | Blinking                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |            |                                        |             |             |           |               |         |          |       |             |        |                      |              |                         |
| A_DIM              | Half bright                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |            |                                        |             |             |           |               |         |          |       |             |        |                      |              |                         |
| A_BOLD             | Extra bright or bold                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |            |                                        |             |             |           |               |         |          |       |             |        |                      |              |                         |
| A_ALTCHARSET       | Alternate character set                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |            |                                        |             |             |           |               |         |          |       |             |        |                      |              |                         |

wstandout(3CURSES)

A\_CHARTEXT Bit-mask to extract a character

COLOR\_PAIR(*n*) Color-pair number *n*

The following macro is the reverse of COLOR\_PAIR(*n*) :

PAIR\_NUMBER(*attrs*) Returns the pair number associated with the COLOR\_PAIR(*n*) attribute

**RETURN VALUES** These routines always return 1.

**ATTRIBUTES** See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level       | Unsafe          |

**SEE ALSO** curs\_addch(3CURSES), curs\_addstr(3CURSES), curs\_printw(3CURSES), curses(3CURSES), attributes(5)

**NOTES** The header < curses . h > automatically includes the headers < stdio . h > and < unctrl . h >.

Note that attroff(), wattroff(), attron(), wattron(), wattrset(), standend(), and standout() may be macros.

| <b>NAME</b>          | standend, standout, wstandend, wstandout – set/clear window attributes                                                                                                                                                                                                                             |                |                 |                     |          |          |        |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|---------------------|----------|----------|--------|
| <b>SYNOPSIS</b>      | <pre>cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [ library... ]  c89 [ flag... ] file... -lcurses [ library... ]  #include &lt;curses.h&gt;  int standend(void);  int standout(void);  int wstandend(WINDOW *win);  int wstandout(WINDOW *win);</pre> |                |                 |                     |          |          |        |
| <b>PARAMETERS</b>    | <p><i>win</i> Is a pointer to the window in which attribute changes are to be made.</p>                                                                                                                                                                                                            |                |                 |                     |          |          |        |
| <b>DESCRIPTION</b>   | <p>The standend() and wstandend() functions turn off all attributes associated with stdscr and <i>win</i> respectively.</p> <p>The standout() and wstandout() functions turn on the A_STANDOUT attribute of stdscr and <i>win</i> respectively.</p>                                                |                |                 |                     |          |          |        |
| <b>RETURN VALUES</b> | These functions always return 1.                                                                                                                                                                                                                                                                   |                |                 |                     |          |          |        |
| <b>ERRORS</b>        | None.                                                                                                                                                                                                                                                                                              |                |                 |                     |          |          |        |
| <b>ATTRIBUTES</b>    | See attributes(5) for descriptions of the following attributes:                                                                                                                                                                                                                                    |                |                 |                     |          |          |        |
|                      | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table>                                                                   | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE       | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                    |                |                 |                     |          |          |        |
| Interface Stability  | Standard                                                                                                                                                                                                                                                                                           |                |                 |                     |          |          |        |
| MT-Level             | Unsafe                                                                                                                                                                                                                                                                                             |                |                 |                     |          |          |        |
| <b>SEE ALSO</b>      | attr_get(3XCURSES), attroff(3XCURSES), libcurses(3XCURSES), attributes(5), standards(5)                                                                                                                                                                                                            |                |                 |                     |          |          |        |

## wsyncdown(3CURSES)

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>NAME</b>        | <code>curs_window</code> , <code>newwin</code> , <code>delwin</code> , <code>mvwin</code> , <code>subwin</code> , <code>derwin</code> , <code>mvderwin</code> , <code>dupwin</code> , <code>wsyncup</code> , <code>syncok</code> , <code>wcursyncup</code> , <code>wsyncdown</code> – create curses windows                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>SYNOPSIS</b>    | <pre>cc [ flag ... ] file ... -lcurses [ library ... ] #include &lt;curses.h&gt;  WINDOW *newwin(int nlines, int ncols, int begin_y, int begin_x); int delwin(WINDOW *win); int mvwin(WINDOW *win, int y, int x); WINDOW *subwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); WINDOW *derwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); int mvderwin(WINDOW *win, int par_y, int par_x); WINDOW *dupwin(WINDOW *win); void wsyncup(WINDOW *win); int syncok(WINDOW *win, bool bf); void wcursyncup(WINDOW *win); void wsyncdown(WINDOW *win);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>DESCRIPTION</b> | <p>The <code>newwin()</code> routine creates and returns a pointer to a new window with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. The upper left-hand corner of the window is at line <i>begin_y</i>, column <i>begin_x</i>. If either <i>nlines</i> or <i>ncols</i> is zero, they default to <code>LINES</code> — <i>begin_y</i> and <code>COLS</code> — <i>begin_x</i>. A new full-screen window is created by calling <code>newwin(0, 0, 0, 0)</code>.</p> <p>The <code>delwin()</code> routine deletes the named window, freeing all memory associated with it. Subwindows must be deleted before the main window can be deleted.</p> <p>The <code>mvwin()</code> routine moves the window so that the upper left-hand corner is at position (<i>x</i>, <i>y</i>). If the move would cause the window to be off the screen, it is an error and the window is not moved. Moving subwindows is allowed, but should be avoided.</p> <p>The <code>subwin()</code> routine creates and returns a pointer to a new window with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. The window is at position (<i>begin_y</i>, <i>begin_x</i>) on the screen. (This position is relative to the screen, and not to the window <i>orig</i>.) The window is made in the middle of the window <i>orig</i>, so that changes made to one window will affect both windows. The subwindow shares memory with the window <i>orig</i>. When using this routine, it is necessary to call <code>touchwin()</code> or <code>touchline()</code> on <i>orig</i> before calling <code>wrefresh()</code> on the subwindow.</p> <p>The <code>derwin()</code> routine is the same as <code>subwin()</code>, except that <i>begin_y</i> and <i>begin_x</i> are relative to the origin of the window <i>orig</i> rather than the screen. There is no difference between the subwindows and the derived windows.</p> |

## wsyncdown(3CURSES)

The `mvderwin()` routine moves a derived window (or subwindow) inside its parent window. The screen-relative parameters of the window are not changed. This routine is used to display different parts of the parent window at the same physical position on the screen.

The `dupwin()` routine creates an exact duplicate of the window *win*.

Each `curses` window maintains two data structures: the character image structure and the status structure. The character image structure is shared among all windows in the window hierarchy (that is, the window with all subwindows). The status structure, which contains information about individual line changes in the window, is private to each window. The routine `wrefresh()` uses the status data structure when performing screen updating. Since status structures are not shared, changes made to one window in the hierarchy may not be properly reflected on the screen.

The routine `wsyncup()` causes the changes in the status structure of a window to be reflected in the status structures of its ancestors. If `syncok()` is called with second argument `TRUE` then `wsyncup()` is called automatically whenever there is a change in the window.

The routine `wcursyncup()` updates the current cursor position of all the ancestors of the window to reflect the current cursor position of the window.

The routine `wsyncdown()` updates the status structure of the window to reflect the changes in the status structures of its ancestors. Applications seldom call this routine because it is called automatically by `wrefresh()`.

### RETURN VALUES

Routines that return an integer return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`delwin()` returns the integer `ERR` upon failure and `OK` upon successful completion.

Routines that return pointers return `NULL` on error.

### ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level       | Unsafe          |

### SEE ALSO

`curs_refresh(3CURSES)`, `curs_touch(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

### NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

If many small changes are made to the window, the `wsyncup()` option could degrade performance.

`wsyncdown(3CURSES)`

Note that `syncok()` may be a macro.

| <b>NAME</b>          | syncok, wcursyncup, wsyncdown, wsyncup – synchronize window with its parents or children                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                |                 |                     |          |          |        |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|---------------------|----------|----------|--------|
| <b>SYNOPSIS</b>      | <pre>cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [ library... ]  c89 [ flag... ] file... -lcurses [ library... ]  #include &lt;curses.h&gt;  int syncok(WINDOW *win, bool bf);  void wcursyncup(WINDOW *win);  void wsyncdown(WINDOW *win);  void wsyncup(WINDOW *win);</pre>                                                                                                                                                                                                                                                                                                                                    |                |                 |                     |          |          |        |
| <b>PARAMETERS</b>    | <p><i>win</i> Is a pointer to a window.</p> <p><i>bf</i> Is a Boolean expression.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                |                 |                     |          |          |        |
| <b>DESCRIPTION</b>   | <p>The <code>syncok()</code> function uses the value of <i>bf</i> to determine whether or not the window <i>win</i>'s ancestors are implicitly touched whenever there is a change to <i>win</i>. If <i>bf</i> is TRUE, this touching occurs. If <i>bf</i> is FALSE, it does not occur. The initial value for <i>bf</i> is FALSE.</p> <p>The <code>wcursyncup()</code> function moves the cursor in <i>win</i>'s ancestors to match its position in <i>win</i>.</p> <p>The <code>wsyncdown()</code> function touches <i>win</i> if any of its ancestors have been touched.</p> <p>The <code>wsyncup()</code> function touches all ancestors of <i>win</i>.</p> |                |                 |                     |          |          |        |
| <b>RETURN VALUES</b> | <p>On success, the <code>syncok()</code> function returns OK. Otherwise, it returns ERR.</p> <p>The other functions do not return a value.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                |                 |                     |          |          |        |
| <b>ERRORS</b>        | None.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                |                 |                     |          |          |        |
| <b>ATTRIBUTES</b>    | See <code>attributes(5)</code> for descriptions of the following attributes:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                |                 |                     |          |          |        |
|                      | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table>                                                                                                                                                                                                                                                                                                                                                                                                                              | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE       | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                |                 |                     |          |          |        |
| Interface Stability  | Standard                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                |                 |                     |          |          |        |
| MT-Level             | Unsafe                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                |                 |                     |          |          |        |
| <b>SEE ALSO</b>      | <code>derwin(3XCURSES)</code> , <code>doupdate(3XCURSES)</code> , <code>is_linetouched(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                |                 |                     |          |          |        |

## wsyncup(3CURSES)

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>NAME</b>        | <code>curs_window</code> , <code>newwin</code> , <code>delwin</code> , <code>mvwin</code> , <code>subwin</code> , <code>derwin</code> , <code>mvderwin</code> , <code>dupwin</code> , <code>wsyncup</code> , <code>syncok</code> , <code>wcursyncup</code> , <code>wsyncdown</code> – create curses windows                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>SYNOPSIS</b>    | <pre>cc [ flag ... ] file ... -lcurses [ library ... ] #include &lt;curses.h&gt;  WINDOW *newwin(int nlines, int ncols, int begin_y, int begin_x); int delwin(WINDOW *win); int mvwin(WINDOW *win, int y, int x); WINDOW *subwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); WINDOW *derwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x); int mvderwin(WINDOW *win, int par_y, int par_x); WINDOW *dupwin(WINDOW *win); void wsyncup(WINDOW *win); int syncok(WINDOW *win, bool bf); void wcursyncup(WINDOW *win); void wsyncdown(WINDOW *win);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>DESCRIPTION</b> | <p>The <code>newwin()</code> routine creates and returns a pointer to a new window with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. The upper left-hand corner of the window is at line <i>begin_y</i>, column <i>begin_x</i>. If either <i>nlines</i> or <i>ncols</i> is zero, they default to <code>LINES</code> — <i>begin_y</i> and <code>COLS</code> — <i>begin_x</i>. A new full-screen window is created by calling <code>newwin(0, 0, 0, 0)</code>.</p> <p>The <code>delwin()</code> routine deletes the named window, freeing all memory associated with it. Subwindows must be deleted before the main window can be deleted.</p> <p>The <code>mvwin()</code> routine moves the window so that the upper left-hand corner is at position (<i>x</i>, <i>y</i>). If the move would cause the window to be off the screen, it is an error and the window is not moved. Moving subwindows is allowed, but should be avoided.</p> <p>The <code>subwin()</code> routine creates and returns a pointer to a new window with the given number of lines, <i>nlines</i>, and columns, <i>ncols</i>. The window is at position (<i>begin_y</i>, <i>begin_x</i>) on the screen. (This position is relative to the screen, and not to the window <i>orig</i>.) The window is made in the middle of the window <i>orig</i>, so that changes made to one window will affect both windows. The subwindow shares memory with the window <i>orig</i>. When using this routine, it is necessary to call <code>touchwin()</code> or <code>touchline()</code> on <i>orig</i> before calling <code>wrefresh()</code> on the subwindow.</p> <p>The <code>derwin()</code> routine is the same as <code>subwin()</code>, except that <i>begin_y</i> and <i>begin_x</i> are relative to the origin of the window <i>orig</i> rather than the screen. There is no difference between the subwindows and the derived windows.</p> |

The `mvderwin()` routine moves a derived window (or subwindow) inside its parent window. The screen-relative parameters of the window are not changed. This routine is used to display different parts of the parent window at the same physical position on the screen.

The `dupwin()` routine creates an exact duplicate of the window *win*.

Each `curses` window maintains two data structures: the character image structure and the status structure. The character image structure is shared among all windows in the window hierarchy (that is, the window with all subwindows). The status structure, which contains information about individual line changes in the window, is private to each window. The routine `wrefresh()` uses the status data structure when performing screen updating. Since status structures are not shared, changes made to one window in the hierarchy may not be properly reflected on the screen.

The routine `wsyncup()` causes the changes in the status structure of a window to be reflected in the status structures of its ancestors. If `syncok()` is called with second argument `TRUE` then `wsyncup()` is called automatically whenever there is a change in the window.

The routine `wcursyncup()` updates the current cursor position of all the ancestors of the window to reflect the current cursor position of the window.

The routine `wsyncdown()` updates the status structure of the window to reflect the changes in the status structures of its ancestors. Applications seldom call this routine because it is called automatically by `wrefresh()`.

**RETURN VALUES**

Routines that return an integer return the integer `ERR` upon failure and an integer value other than `ERR` upon successful completion.

`delwin()` returns the integer `ERR` upon failure and `OK` upon successful completion.

Routines that return pointers return `NULL` on error.

**ATTRIBUTES**

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level       | Unsafe          |

**SEE ALSO**

`curs_refresh(3CURSES)`, `curs_touch(3CURSES)`, `curses(3CURSES)`, `attributes(5)`

**NOTES**

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

If many small changes are made to the window, the `wsyncup()` option could degrade performance.

`wsyncup(3CURSES)`

Note that `syncok()` may be a macro.

| <b>NAME</b>          | syncok, wcuryncup, wsyncdown, wsyncup – synchronize window with its parents or children                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                |                 |                     |          |          |        |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|---------------------|----------|----------|--------|
| <b>SYNOPSIS</b>      | <pre>cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [ library... ]  c89 [ flag... ] file... -lcurses [ library... ]  #include &lt;curses.h&gt;  int syncok(WINDOW *win, bool bf);  void wcuryncup(WINDOW *win);  void wsyncdown(WINDOW *win);  void wsyncup(WINDOW *win);</pre>                                                                                                                                                                                                                                                                                                                                    |                |                 |                     |          |          |        |
| <b>PARAMETERS</b>    | <p><i>win</i> Is a pointer to a window.</p> <p><i>bf</i> Is a Boolean expression.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                |                 |                     |          |          |        |
| <b>DESCRIPTION</b>   | <p>The <code>syncok()</code> function uses the value of <i>bf</i> to determine whether or not the window <i>win</i>'s ancestors are implicitly touched whenever there is a change to <i>win</i>. If <i>bf</i> is TRUE, this touching occurs. If <i>bf</i> is FALSE, it does not occur. The initial value for <i>bf</i> is FALSE.</p> <p>The <code>wcuryncup()</code> function moves the cursor in <i>win</i>'s ancestors to match its position in <i>win</i>.</p> <p>The <code>wsyncdown()</code> function touches <i>win</i> if any of its ancestors have been touched.</p> <p>The <code>wsyncup()</code> function touches all ancestors of <i>win</i>.</p> |                |                 |                     |          |          |        |
| <b>RETURN VALUES</b> | <p>On success, the <code>syncok()</code> function returns OK. Otherwise, it returns ERR.</p> <p>The other functions do not return a value.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                |                 |                     |          |          |        |
| <b>ERRORS</b>        | None.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                |                 |                     |          |          |        |
| <b>ATTRIBUTES</b>    | See <code>attributes(5)</code> for descriptions of the following attributes:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                |                 |                     |          |          |        |
|                      | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Standard</td> </tr> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table>                                                                                                                                                                                                                                                                                                                                                                                                                             | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Standard | MT-Level | Unsafe |
| ATTRIBUTE TYPE       | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                |                 |                     |          |          |        |
| Interface Stability  | Standard                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                |                 |                     |          |          |        |
| MT-Level             | Unsafe                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                |                 |                     |          |          |        |
| <b>SEE ALSO</b>      | <code>derwin(3XCURSES)</code> , <code>doupdate(3XCURSES)</code> , <code>is_linetouched(3XCURSES)</code> , <code>libcurses(3XCURSES)</code> , <code>attributes(5)</code> , <code>standards(5)</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                |                 |                     |          |          |        |

wtimeout(3CURSES)

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>NAME</b>        | <code>curs_inopts</code> , <code>cbreak</code> , <code>nocbreak</code> , <code>echo</code> , <code>noecho</code> , <code>halfdelay</code> , <code>intrflush</code> , <code>keypad</code> , <code>meta</code> , <code>nodelay</code> , <code>notimeout</code> , <code>raw</code> , <code>noraw</code> , <code>noqiflush</code> , <code>qiflush</code> , <code>timeout</code> , <code>wtimeout</code> , <code>typeahead</code> – curses terminal input option control routines                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>SYNOPSIS</b>    | <pre>cc [ <i>flag</i> ... ] <i>file</i> ... -lcurses [ <i>library</i> ... ] #include &lt;curses.h&gt;  int <b>cbreak</b>(void); int <b>nocbreak</b>(void); int <b>echo</b>(void); int <b>noecho</b>(void); int <b>halfdelay</b>(int <i>tenths</i>); int <b>intrflush</b>(WINDOW *<i>win</i>, bool <i>bf</i>); int <b>keypad</b>(WINDOW *<i>win</i>, bool <i>bf</i>); int <b>meta</b>(WINDOW *<i>win</i>, bool <i>bf</i>); int <b>nodelay</b>(WINDOW *<i>win</i>, bool <i>bf</i>); int <b>notimeout</b>(WINDOW *<i>win</i>, bool <i>bf</i>); int <b>raw</b>(void); int <b>noraw</b>(void); void <b>noqiflush</b>(void); void <b>qiflush</b>(void); void <b>timeout</b>(int <i>delay</i>); void <b>wtimeout</b>(WINDOW *<i>win</i>, int <i>delay</i>); int <b>typeahead</b>(int <i>fildes</i>);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>DESCRIPTION</b> | <p>The <code>cbreak()</code> and <code>nocbreak()</code> routines put the terminal into and out of <code>cbreak()</code> mode, respectively. In this mode, characters typed by the user are immediately available to the program, and erase/kill character-processing is not performed. When out of this mode, the tty driver buffers the typed characters until a newline or carriage return is typed. Interrupt and flow control characters are unaffected by this mode. Initially the terminal may or may not be in <code>cbreak()</code> mode, as the mode is inherited; therefore, a program should call <code>cbreak()</code> or <code>nocbreak()</code> explicitly. Most interactive programs using curses set the <code>cbreak()</code> mode.</p> <p>Note that <code>cbreak()</code> overrides <code>raw()</code>. (See <code>curs_getch(3CURSES)</code> for a discussion of how these routines interact with <code>echo()</code> and <code>noecho()</code>.)</p> <p>The <code>echo()</code> and <code>noecho()</code> routines control whether characters typed by the user are echoed by <code>getch()</code> as they are typed. Echoing by the tty driver is always disabled, but initially <code>getch()</code> is in echo mode, so characters typed are echoed. Authors of</p> |

most interactive programs prefer to do their own echoing in a controlled area of the screen, or not to echo at all, so they disable echoing by calling `noecho()`. (See `curs_getch(3CURSES)` for a discussion of how these routines interact with `cbreak()` and `nocbreak()`.)

The `halfdelay()` routine is used for half-delay mode, which is similar to `cbreak()` mode in that characters typed by the user are immediately available to the program. However, after blocking for *tenths* tenths of seconds, `ERR` is returned if nothing has been typed. The value of *tenths* must be a number between 1 and 255. Use `nocbreak()` to leave half-delay mode.

If the `intrflush()` option is enabled, (*bf* is `TRUE`), when an interrupt key is pressed on the keyboard (interrupt, break, quit) all output in the tty driver queue will be flushed, giving the effect of faster response to the interrupt, but causing `curses` to have the wrong idea of what is on the screen. Disabling (*bf* is `FALSE`), the option prevents the flush. The default for the option is inherited from the tty driver settings. The window argument is ignored.

The `keypad()` option enables the keypad of the user's terminal. If enabled (*bf* is `TRUE`), the user can press a function key (such as an arrow key) and `wgetch()` returns a single value representing the function key, as in `KEY_LEFT`. If disabled (*bf* is `FALSE`), `curses` does not treat function keys specially and the program has to interpret the escape sequences itself. If the keypad in the terminal can be turned on (made to transmit) and off (made to work locally), turning on this option causes the terminal keypad to be turned on when `wgetch()` is called. The default value for keypad is false.

Initially, whether the terminal returns 7 or 8 significant bits on input depends on the control mode of the tty driver (see `termio(7I)`). To force 8 bits to be returned, invoke `meta(win, TRUE)`. To force 7 bits to be returned, invoke `meta(win, FALSE)`. The window argument, *win*, is always ignored. If the terminfo capabilities `smm` (`meta_on`) and `rmm` (`meta_off`) are defined for the terminal, `smm` is sent to the terminal when `meta(win, TRUE)` is called and `rmm` is sent when `meta(win, FALSE)` is called.

The `nodelay()` option causes `getch()` to be a non-blocking call. If no input is ready, `getch()` returns `ERR`. If disabled (*bf* is `FALSE`), `getch()` waits until a key is pressed.

While interpreting an input escape sequence, `wgetch()` sets a timer while waiting for the next character. If `notimeout(win, TRUE)` is called, then `wgetch()` does not set a timer. The purpose of the timeout is to differentiate between sequences received from a function key and those typed by a user.

With the `raw()` and `noraw()` routines, the terminal is placed into or out of raw mode. Raw mode is similar to `cbreak()` mode, in that characters typed are immediately passed through to the user program. The differences are that in raw mode, the interrupt, quit, suspend, and flow control characters are all passed through uninterpreted, instead of generating a signal. The behavior of the `BREAK` key depends on other bits in the tty driver that are not set by `curses`.

## wtimeout(3CURSES)

When the `noqiflush()` routine is used, normal flush of input and output queues associated with the `INTR`, `QUIT` and `SUSP` characters will not be done (see `termio(7I)`). When `qiflush()` is called, the queues will be flushed when these control characters are read.

The `timeout()` and `wtimeout()` routines set blocking or non-blocking read for a given window. If *delay* is negative, blocking read is used (that is, waits indefinitely for input). If *delay* is zero, then non-blocking read is used (that is, read returns `ERR` if no input is waiting). If *delay* is positive, then read blocks for *delay* milliseconds, and returns `ERR` if there is still no input. Hence, these routines provide the same functionality as `nodelay()`, plus the additional capability of being able to block for only *delay* milliseconds (where *delay* is positive).

`curses` does "line-breakout optimization" by looking for typeahead periodically while updating the screen. If input is found, and it is coming from a tty, the current update is postponed until `refresh()` or `doupdate()` is called again. This allows faster response to commands typed in advance. Normally, the input FILE pointer passed to `newterm()`, or `stdin` in the case that `initscr()` was used, will be used to do this typeahead checking. The `typeahead()` routine specifies that the file descriptor *fildev* is to be used to check for typeahead instead. If *fildev* is `-1`, then no typeahead checking is done.

### RETURN VALUES

All routines that return an integer return `ERR` upon failure and an integer value other than `ERR` upon successful completion, unless otherwise noted in the preceding routine descriptions.

### ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level       | Unsafe          |

### SEE ALSO

`curs_getch(3CURSES)`, `curs_initscr(3CURSES)`, `curses(3CURSES)`, `attributes(5)`, `termio(7I)`

### NOTES

The header `<curses.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `echo()`, `noecho()`, `halfdelay()`, `intrflush()`, `meta()`, `nodelay()`, `notimeout()`, `noqiflush()`, `qiflush()`, `timeout()`, and `wtimeout()` may be macros.

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |            |                                                                |           |                          |              |                                                           |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|----------------------------------------------------------------|-----------|--------------------------|--------------|-----------------------------------------------------------|
| <b>NAME</b>          | notimeout, timeout, wtimeout – set timed blocking or non-blocking read                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |            |                                                                |           |                          |              |                                                           |
| <b>SYNOPSIS</b>      | <pre>cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [ library... ]  c89 [ flag... ] file... -lcurses [ library... ]  #include &lt;curses.h&gt;  int notimeout(WINDOW *win, bool bf);  void timeout(int delay);  void wtimeout(WINDOW win, int delay);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |            |                                                                |           |                          |              |                                                           |
| <b>PARAMETERS</b>    | <table border="0"> <tr> <td style="padding-right: 20px;"><i>win</i></td> <td>Is a pointer to the window in which to set the timed blocking.</td> </tr> <tr> <td><i>bf</i></td> <td>Is a Boolean expression.</td> </tr> <tr> <td><i>delay</i></td> <td>Is the number of milliseconds to block or wait for input.</td> </tr> </table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | <i>win</i> | Is a pointer to the window in which to set the timed blocking. | <i>bf</i> | Is a Boolean expression. | <i>delay</i> | Is the number of milliseconds to block or wait for input. |
| <i>win</i>           | Is a pointer to the window in which to set the timed blocking.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |            |                                                                |           |                          |              |                                                           |
| <i>bf</i>            | Is a Boolean expression.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |            |                                                                |           |                          |              |                                                           |
| <i>delay</i>         | Is the number of milliseconds to block or wait for input.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |            |                                                                |           |                          |              |                                                           |
| <b>DESCRIPTION</b>   | <p>If <i>bool</i> is TRUE, the notimeout() function disables a timer used by getch(3XCURSES) when handling multibyte function key sequences.</p> <p>When <i>bool</i> is FALSE and keypad handling is enabled, a timer is set by getch() to handle bytes received that could be the beginning of a function key (for example, ESC). If the remainder of the sequence is not received before the time expires, the first byte is returned; otherwise, the value of the function key is returned. Subsequent calls to the getch() function will return the other bytes received for the incomplete key sequence.</p> <p>The timeout() and wtimeout() functions set the length of time getch() waits for input for windows stdscr and <i>win</i>, respectively. These functions are similar to nodelay(3XCURSES) except the time to block or wait for input can be specified.</p> <p>A negative <i>delay</i> causes the program to wait indefinitely for input; a <i>delay</i> of 0 returns ERR if no input is ready; and a positive <i>delay</i> blocks until input arrives or the time specified expires, (in which case, ERR is returned).</p> |            |                                                                |           |                          |              |                                                           |
| <b>RETURN VALUES</b> | <p>On success, the notimeout() function returns OK. Otherwise, it returns ERR.</p> <p>The timeout() and wtimeout() functions do not return a value.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |            |                                                                |           |                          |              |                                                           |
| <b>ERRORS</b>        | None.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |            |                                                                |           |                          |              |                                                           |
| <b>ATTRIBUTES</b>    | See attributes(5) for descriptions of the following attributes:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |            |                                                                |           |                          |              |                                                           |

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard        |

wtimeout(3XCURSES)

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level       | Unsafe          |

**SEE ALSO** `getch(3XCURSES)`, `halfdelay(3XCURSES)`, `libcurses(3XCURSES)`,  
`nodelay(3XCURSES)`, `attributes(5)`, `standards(5)`

| <b>NAME</b>          | curs_touch, touchwin, touchline, untouchwin, wtouchln, is_linetouched, is_wintouched – curses refresh control routines                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                |                 |          |        |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|----------|--------|
| <b>SYNOPSIS</b>      | <pre>cc [ flag ... ] file ... -lcurses [ library ... ] #include &lt;curses.h&gt;  int touchwin(WINDOW *win); int touchline(WINDOW *win, int start, int count); int untouchwin(WINDOW *win); int wtouchln(WINDOW *win, int y, int n, int changed); int is_linetouched(WINDOW *win, int line); int is_wintouched(WINDOW *win);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                |                 |          |        |
| <b>DESCRIPTION</b>   | <p>The touchwin() and touchline() routines throw away all optimization information about which parts of the window have been touched, by pretending that the entire window has been drawn on. This is sometimes necessary when using overlapping windows, since a change to one window affects the other window, but the records of which lines have been changed in the other window do not reflect the change. The routine touchline() only pretends that <i>count</i> lines have been changed, beginning with line <i>start</i>.</p> <p>The untouchwin() routine marks all lines in the window as unchanged since the last call to wrefresh().</p> <p>The wtouchln() routine makes <i>n</i> lines in the window, starting at line <i>y</i>, look as if they have (<i>changed</i>=1) or have not (<i>changed</i>=0) been changed since the last call to wrefresh().</p> <p>The is_linetouched() and is_wintouched() routines return TRUE if the specified line/window was modified since the last call to wrefresh(); otherwise they return FALSE. In addition, is_linetouched() returns ERR if line is not valid for the given window.</p> |                |                 |          |        |
| <b>RETURN VALUES</b> | All routines return the integer ERR upon failure and an integer value other than ERR upon successful completion, unless otherwise noted in the preceding routine descriptions.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                |                 |          |        |
| <b>ATTRIBUTES</b>    | See attributes(5) for descriptions of the following attributes:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                |                 |          |        |
|                      | <table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>MT-Level</td> <td>Unsafe</td> </tr> </tbody> </table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | ATTRIBUTE TYPE | ATTRIBUTE VALUE | MT-Level | Unsafe |
| ATTRIBUTE TYPE       | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                |                 |          |        |
| MT-Level             | Unsafe                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                |                 |          |        |
| <b>SEE ALSO</b>      | curs_refresh(3CURSES), curses(3CURSES), attributes(5)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                |                 |          |        |
| <b>NOTES</b>         | The header <curses.h> automatically includes the headers <stdio.h> and <unctrl.h>.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                |                 |          |        |

`wtouchln(3CURSES)`

Note that all routines except `wtouchln()` may be macros.

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |            |                                                                                   |             |                                                     |              |                                                                                  |              |                                                          |          |                                                                                                 |          |                                                          |                |                                                                                 |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|-----------------------------------------------------------------------------------|-------------|-----------------------------------------------------|--------------|----------------------------------------------------------------------------------|--------------|----------------------------------------------------------|----------|-------------------------------------------------------------------------------------------------|----------|----------------------------------------------------------|----------------|---------------------------------------------------------------------------------|
| <b>NAME</b>        | is_linetouched, is_wintouched, touchline, touchwin, untouchwin, wtouchln – control window refresh                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |            |                                                                                   |             |                                                     |              |                                                                                  |              |                                                          |          |                                                                                                 |          |                                                          |                |                                                                                 |
| <b>SYNOPSIS</b>    | <pre>cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [ library... ]  c89 [ flag... ] file... -lcurses [ library... ]  #include &lt;curses.h&gt;  bool is_linetouched(WINDOW *win, int line); bool is_wintouched(WINDOW *win); int touchline(WINDOW *win, int start, int count); int touchwin(WINDOW *win); int untouchwin(WINDOW *win); int wtouchln(WINDOW *win, int y, int n, int changed);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                   |            |                                                                                   |             |                                                     |              |                                                                                  |              |                                                          |          |                                                                                                 |          |                                                          |                |                                                                                 |
| <b>PARAMETERS</b>  | <table border="0"> <tr> <td style="padding-right: 20px;"><i>win</i></td> <td>Is a pointer to the window in which the refresh is to be controlled or monitored.</td> </tr> <tr> <td><i>line</i></td> <td>Is the line to be checked for change since refresh.</td> </tr> <tr> <td><i>start</i></td> <td>Is the starting line number of the portion of the window to make appear changed.</td> </tr> <tr> <td><i>count</i></td> <td>Is the number of lines in the window to mark as changed.</td> </tr> <tr> <td><i>y</i></td> <td>Is the starting line number of the portion of the window to make appear changed or not changed.</td> </tr> <tr> <td><i>n</i></td> <td>Is the number of lines in the window to mark as changed.</td> </tr> <tr> <td><i>changed</i></td> <td>Is a flag indicating whether to make lines look changed (0) or not changed (1).</td> </tr> </table> | <i>win</i> | Is a pointer to the window in which the refresh is to be controlled or monitored. | <i>line</i> | Is the line to be checked for change since refresh. | <i>start</i> | Is the starting line number of the portion of the window to make appear changed. | <i>count</i> | Is the number of lines in the window to mark as changed. | <i>y</i> | Is the starting line number of the portion of the window to make appear changed or not changed. | <i>n</i> | Is the number of lines in the window to mark as changed. | <i>changed</i> | Is a flag indicating whether to make lines look changed (0) or not changed (1). |
| <i>win</i>         | Is a pointer to the window in which the refresh is to be controlled or monitored.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |            |                                                                                   |             |                                                     |              |                                                                                  |              |                                                          |          |                                                                                                 |          |                                                          |                |                                                                                 |
| <i>line</i>        | Is the line to be checked for change since refresh.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |            |                                                                                   |             |                                                     |              |                                                                                  |              |                                                          |          |                                                                                                 |          |                                                          |                |                                                                                 |
| <i>start</i>       | Is the starting line number of the portion of the window to make appear changed.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |            |                                                                                   |             |                                                     |              |                                                                                  |              |                                                          |          |                                                                                                 |          |                                                          |                |                                                                                 |
| <i>count</i>       | Is the number of lines in the window to mark as changed.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |            |                                                                                   |             |                                                     |              |                                                                                  |              |                                                          |          |                                                                                                 |          |                                                          |                |                                                                                 |
| <i>y</i>           | Is the starting line number of the portion of the window to make appear changed or not changed.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |            |                                                                                   |             |                                                     |              |                                                                                  |              |                                                          |          |                                                                                                 |          |                                                          |                |                                                                                 |
| <i>n</i>           | Is the number of lines in the window to mark as changed.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |            |                                                                                   |             |                                                     |              |                                                                                  |              |                                                          |          |                                                                                                 |          |                                                          |                |                                                                                 |
| <i>changed</i>     | Is a flag indicating whether to make lines look changed (0) or not changed (1).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |            |                                                                                   |             |                                                     |              |                                                                                  |              |                                                          |          |                                                                                                 |          |                                                          |                |                                                                                 |
| <b>DESCRIPTION</b> | <p>The touchwin() function marks the entire window as dirty. This makes it appear to X/Open Curses as if the whole window has been changed, thus causing the entire window to be rewritten with the next call to refresh(3XCURSES). This is sometimes necessary when using overlapping windows; the change to one window will not be reflected in the other and, hence will not be recorded.</p> <p>The touchline() function marks as dirty a portion of the window starting at line <i>start</i> and continuing for <i>count</i> lines instead of the entire window. Consequently, that portion of the window is updated with the next call to refresh().</p> <p>The untouchwin() function marks all lines in the window as unchanged since the last refresh, ensuring that it is not updated.</p>                                                                            |            |                                                                                   |             |                                                     |              |                                                                                  |              |                                                          |          |                                                                                                 |          |                                                          |                |                                                                                 |

## wtouchln(3XCURSES)

The `wtouchln()` function marks  $n$  lines starting at line  $y$  as either changed (*changed=1*) or unchanged (*changed=0*) since the last refresh.

To find out which lines or windows have been changed since the last refresh, use the `is_linetouched()` and `is_wintouched()` functions, respectively. These return `TRUE` if the specified line or window have been changed since the last call to `refresh()` or `FALSE` if no changes have been made.

**RETURN VALUES** On success, these functions return `OK`. Otherwise, they return `ERR`.

**ERRORS** None.

**ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard        |
| MT-Level            | Unsafe          |

**SEE ALSO** `doupdate(3XCURSES)`, `libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

**NAME** wunctrl – generate printable representation of a wide character

**SYNOPSIS**

```
cc [flag...] file... -I /usr/xpg4/include -L /usr/xpg4/lib \
-R /usr/xpg4/lib -lcurses [library...]

c89 [flag...] file... -lcurses [library...]

#include <curses.h>

wchar_t *wunctrl (cchar_t *wc);
```

**PARAMETERS** *wc* Is a pointer to the wide character.

**DESCRIPTION** The wunctrl() function converts the a wide character string that is a printable representation of the wide character *wc*.

This function also performs the following processing on the input argument:

- Control characters are converted to the ^X notation
- Any rendition information is removed.

**RETURN VALUES** Upon successful completion, the wunctrl() function returns the generated string. Otherwise, it returns a null pointer.

**ERRORS** No errors are defined.

**ATTRIBUTES** See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard        |
| MT-Level            | Unsafe          |

**SEE ALSO** keyname(3XCURSES), libcurses(3XCURSES), unctrl(3XCURSES), attributes(5), standards(5)

## wvline(3CURSES)

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                 |                         |                 |                          |                 |                        |                 |                           |                 |                      |                 |                       |                 |                         |                 |                          |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-------------------------|-----------------|--------------------------|-----------------|------------------------|-----------------|---------------------------|-----------------|----------------------|-----------------|-----------------------|-----------------|-------------------------|-----------------|--------------------------|
| <b>NAME</b>        | <code> curs_border, border, wborder, box, whline, wvline</code> – create curses borders, horizontal and vertical lines                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                 |                         |                 |                          |                 |                        |                 |                           |                 |                      |                 |                       |                 |                         |                 |                          |
| <b>SYNOPSIS</b>    | <pre>cc [ flag ... ] file ... -lcurses [ library ... ] #include &lt;curses.h&gt;  int <b>border</b>(chtype <i>ls</i>, chtype <i>rs</i>, chtype <i>ts</i>, chtype <i>bs</i>, chtype <i>tl</i>,            chtype <i>tr</i>, chtype <i>bl</i>, chtype <i>br</i>);  int <b>wborder</b>(WINDOW *<i>win</i>, chtype <i>ls</i>, chtype <i>rs</i>, chtype <i>ts</i>, chtype <i>bs</i>,            chtype <i>tl</i>, chtype <i>tr</i>, chtype <i>bl</i>, chtype <i>br</i>);  int <b>box</b>(WINDOW *<i>win</i>, chtype <i>verch</i>, chtype <i>horch</i>);  int <b>hline</b>(chtype <i>ch</i>, int <i>n</i>);  int <b>whline</b>(WINDOW *<i>win</i>, chtype <i>ch</i>, int <i>n</i>);  int <b>vline</b>(chtype <i>ch</i>, int <i>n</i>);  int <b>wvline</b>(WINDOW *<i>win</i>, chtype <i>ch</i>, int <i>n</i>);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                 |                         |                 |                          |                 |                        |                 |                           |                 |                      |                 |                       |                 |                         |                 |                          |
| <b>DESCRIPTION</b> | <p>With the <code>border()</code>, <code>wborder()</code>, and <code>box()</code> routines, a border is drawn around the edges of the window. The arguments and attributes are:</p> <hr/> <table><tr><td><code>ls</code></td><td>left side of the border</td></tr><tr><td><code>rs</code></td><td>right side of the border</td></tr><tr><td><code>ts</code></td><td>top side of the border</td></tr><tr><td><code>bs</code></td><td>bottom side of the border</td></tr><tr><td><code>tl</code></td><td>top left-hand corner</td></tr><tr><td><code>tr</code></td><td>top right-hand corner</td></tr><tr><td><code>bl</code></td><td>bottom left-hand corner</td></tr><tr><td><code>br</code></td><td>bottom right-hand corner</td></tr></table> <hr/> <p>If any of these arguments is zero, then the following default values (defined in <code>&lt;curses.h&gt;</code>) are used respectively instead: <code>ACS_VLINE</code>, <code>ACS_VLINE</code>, <code>ACS_HLINE</code>, <code>ACS_HLINE</code>, <code>ACS_ULCORNER</code>, <code>ACS_URCORNER</code>, <code>ACS_BLCORNER</code>, <code>ACS_BRCORNER</code>.</p> <p><code>box(win, verch, horch)</code> is a shorthand for the following call:</p> <pre>wborder(win, verch, verch, horch, horch, 0, 0, 0, 0)</pre> <p><code>hline()</code> and <code>whline()</code> draw a horizontal (left to right) line using <code>ch</code> starting at the current cursor position in the window. The current cursor position is not changed. The line is at most <code>n</code> characters long, or as many as fit into the window.</p> | <code>ls</code> | left side of the border | <code>rs</code> | right side of the border | <code>ts</code> | top side of the border | <code>bs</code> | bottom side of the border | <code>tl</code> | top left-hand corner | <code>tr</code> | top right-hand corner | <code>bl</code> | bottom left-hand corner | <code>br</code> | bottom right-hand corner |
| <code>ls</code>    | left side of the border                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                 |                         |                 |                          |                 |                        |                 |                           |                 |                      |                 |                       |                 |                         |                 |                          |
| <code>rs</code>    | right side of the border                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                 |                         |                 |                          |                 |                        |                 |                           |                 |                      |                 |                       |                 |                         |                 |                          |
| <code>ts</code>    | top side of the border                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                 |                         |                 |                          |                 |                        |                 |                           |                 |                      |                 |                       |                 |                         |                 |                          |
| <code>bs</code>    | bottom side of the border                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                 |                         |                 |                          |                 |                        |                 |                           |                 |                      |                 |                       |                 |                         |                 |                          |
| <code>tl</code>    | top left-hand corner                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                 |                         |                 |                          |                 |                        |                 |                           |                 |                      |                 |                       |                 |                         |                 |                          |
| <code>tr</code>    | top right-hand corner                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                 |                         |                 |                          |                 |                        |                 |                           |                 |                      |                 |                       |                 |                         |                 |                          |
| <code>bl</code>    | bottom left-hand corner                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                 |                         |                 |                          |                 |                        |                 |                           |                 |                      |                 |                       |                 |                         |                 |                          |
| <code>br</code>    | bottom right-hand corner                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                 |                         |                 |                          |                 |                        |                 |                           |                 |                      |                 |                       |                 |                         |                 |                          |

wvline(3CURSES)

vline() and wvline() draw a vertical (top to bottom) line using *ch* starting at the current cursor position in the window. The current cursor position is not changed. The line is at most *n* characters long, or as many as fit into the window.

**RETURN VALUES** All routines return the integer OK, or a non-negative integer if `immedok()` is set. See `curl_oulopts(3CURSES)`.

**ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| MT-Level       | Unsafe          |

**SEE ALSO** `curl_oulopts(3CURSES)`, `curls(3CURSES)`, `attributes(5)`

**NOTES** The header `<curls.h>` automatically includes the headers `<stdio.h>` and `<unctrl.h>`.

Note that `border()` and `box()` may be macros.

## wvline(3XCURSES)

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>NAME</b>          | <code>hline</code> , <code>mvhline</code> , <code>mvvline</code> , <code>mvwhline</code> , <code>mvwvline</code> , <code>vline</code> , <code>whline</code> , <code>wvline</code> – use single-byte characters (and renditions) to draw lines                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>SYNOPSIS</b>      | <pre>cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [ library... ]  c89 [ flag... ] file... -lcurses [ library... ]  #include &lt;curses.h&gt;  int hline(chtype ch, int n); int mvhline(int y, int x, chtype ch, int n); int mvvline(int y, int x, chtype ch, int n); int mvwhline(WINDOW *win, int y, int x, chtype ch, int n); int mvwvline(WINDOW *win, int y, int x, chtype ch, int n); int vline(chtype ch, int n); int whline(WINDOW *win, chtype ch, int n); int wvline(WINDOW *win, chtype ch, int n);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>DESCRIPTION</b>   | <p>The <code>hline()</code>, <code>vline()</code>, <code>whline()</code>, <code>wvline()</code> functions draw a horizontal or vertical line, in either the window <code>stdscr</code> or <code>win</code> starting at the current cursor position. The line is drawn using the character <code>ch</code> and is a maximum of <code>n</code> positions long, or as many as will fit into the window. If <code>ch</code> is 0 (zero), the default horizontal or vertical character is used.</p> <p>The <code>mvhline()</code>, <code>mvvline()</code>, <code>mvwhline()</code>, <code>mvwvline()</code> functions are similar to the previous group of functions but the line begins at cursor position specified by <code>x</code> and <code>y</code>.</p> <p>The functions with names ending with <code>hline()</code> draw horizontal lines proceeding towards the last column of the same line. The functions with names ending with <code>vline()</code> draw vertical lines proceeding towards the last column of the same line.</p> <p>These functions do not change the position of the cursor.</p> |
| <b>PARAMETERS</b>    | <p><code>ch</code> Is the character used to draw the line.</p> <p><code>n</code> Is the maximum number of characters in the line.</p> <p><code>y</code> Is the <code>y</code> (row) coordinate for the start of the line.</p> <p><code>x</code> Is the <code>x</code> (column) coordinate for the start of the line.</p> <p><code>win</code> Is a pointer to a window.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>RETURN VALUES</b> | On success, these functions return OK. Otherwise, they return ERR.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>ERRORS</b>        | None                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

wvline(3XCURSES)

**ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard        |
| MT-Level            | Unsafe          |

**SEE ALSO** `border(3XCURSES)`, `border_set(3XCURSES)`, `hline_set(3XCURSES)`,  
`libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

## wvline\_set(3XCURSES)

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>NAME</b>          | <code>hline_set</code> , <code>mvhline_set</code> , <code>mvvline_set</code> , <code>mvwhline_set</code> , <code>mvwvline_set</code> , <code>vline_set</code> , <code>whline_set</code> , <code>wvline_set</code> – use complex characters (and renditions) to draw lines                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>SYNOPSIS</b>      | <pre>cc [ flag... ] file... -I /usr/xpg4/include -L /usr/xpg4/lib \ -R /usr/xpg4/lib -lcurses [ library... ]  c89 [ flag... ] file... -lcurses [ library... ]  #include &lt;curses.h&gt;  int hline_set(const cchar_t *ch, int n);  int mvhline_set(int y, int x, const cchar_t *wch, int n);  int mvvline_set(int y, int x, const cchar_t *wch, int n);  int mvwhline_set(WINDOW *win, int y, int x, const cchar_t *wch, int n);  int mvwvline_set(WINDOW *win, int y, int x, const cchar_t *wch, int n);  int vline_set(const cchar_t *wch, int n);  int whline_set(WINDOW *win, const cchar_t *wch, int n);  int wvline_set(WINDOW *win, const cchar_t *wch, int n);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>DESCRIPTION</b>   | <p>The <code>hline_set()</code>, <code>vline_set()</code>, <code>whline_set()</code>, <code>wvline_set()</code> functions draw a line, in either the window <code>stdscr</code> or <code>win</code> starting at the current cursor position. The line is drawn using the character <code>wch</code> and is a maximum of <code>n</code> positions long, or as many as will fit into the window. If <code>wch</code> is a null pointer, the default horizontal or vertical character is used.</p> <p>The <code>mvhline_set()</code>, <code>mvvline_set()</code>, <code>mvwhline_set()</code>, <code>mvwvline_set()</code> functions are similar to the previous group of functions but the line begins at cursor position specified by <code>x</code> and <code>y</code>.</p> <p>The functions with names ending with <code>hline_set()</code> draw horizontal lines proceeding towards the last column of the same line. The functions with names ending with <code>vline_set()</code> draw vertical lines proceeding towards the last column of the same line.</p> <p>These functions do not change the position of the cursor.</p> |
| <b>PARAMETERS</b>    | <p><code>wch</code> Is the complex character used to draw the line.</p> <p><code>n</code> Is the maximum number of characters in the line.</p> <p><code>y</code> Is the y (row) coordinate for the start of the line.</p> <p><code>x</code> Is the x (column) coordinate for the start of the line.</p> <p><code>win</code> Is a pointer to a window.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>RETURN VALUES</b> | On success, these functions return OK. Otherwise, they return ERR.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

wvline\_set(3XCURSES)

**ERRORS** None.

**ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard        |
| MT-Level            | Unsafe          |

**ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard        |
| MT-Level            | Unsafe          |

**ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Standard        |
| MT-Level            | Unsafe          |

**SEE ALSO** `border(3XCURSES)`, `border_set(3XCURSES)`, `hline(3XCURSES)`,  
`libcurses(3XCURSES)`, `attributes(5)`, `standards(5)`

wvline\_set(3XCURSES)

# Index

---

## A

- activate audio-visual alarm
  - beep, 100
  - flash, 100
- add a string of `wchar_t` characters to a curses window and advance cursor —
  - `curs_addwstr`, 199
- add a `wchar_t` character (with attributes) to a curses window and advance cursor —
  - `curs_addwch`, 194
- add string of `wchar_t` characters (and attributes) to a curses window —  `curs_addwchstr`, 197
- add a string of `wchar_t` characters to a curses window and advance cursor —
  - `curs_addwstr`
  - `addnwstr`, 199
- add a `wchar_t` character (with attributes) to a curses window and advance cursor —
  - `curs_addwch`
  - `addwch`, 194
- add string of `wchar_t` characters (and attributes) to a curses window —  `curs_addwchstr`
  - `addwchnstr`, 197
  - `addwchstr`, 197
- add a string of `wchar_t` characters to a curses window and advance cursor —
  - `curs_addwstr`
  - `addwstr`, 199
- add a `wchar_t` character (with attributes) to a curses window and advance cursor —
  - `curs_addwch`
  - `echowchar`, 194
- add a string of `wchar_t` characters to a curses window and advance cursor —
  - `curs_addwstr`
  - `mvaddnwstr`, 199
- add a `wchar_t` character (with attributes) to a curses window and advance cursor —
  - `curs_addwch`
  - `mvaddwch`, 194
- add string of `wchar_t` characters (and attributes) to a curses window —  `curs_addwchstr`
  - `mvaddwchnstr`, 197
  - `mvaddwchstr`, 197
- add a string of `wchar_t` characters to a curses window and advance cursor —
  - `curs_addwstr`
  - `mvaddwstr`, 199
  - `mvwaddnwstr`, 199
- add a `wchar_t` character (with attributes) to a curses window and advance cursor —
  - `curs_addwch`
  - `mvwaddwch`, 194
- add string of `wchar_t` characters (and attributes) to a curses window —  `curs_addwchstr`
  - `mvwaddwchnstr`, 197
  - `mvwaddwchstr`, 197
- add a string of `wchar_t` characters to a curses window and advance cursor —
  - `curs_addwstr`
  - `mvwaddwstr`, 199
  - `waddnwstr`, 199
- add a `wchar_t` character (with attributes) to a curses window and advance cursor —
  - `curs_addwch`

add a `wchar_t` character (with attributes) to a curses window and advance cursor — `curs_addwch` (Continued)  
     `waddwch`, 194  
 add string of `wchar_t` characters (and attributes) to a curses window — `curs_addwchstr`  
     `waddwchnstr`, 197  
     `waddwchstr`, 197  
 add a string of `wchar_t` characters to a curses window and advance cursor — `curs_addwstr`  
     `waddwstr`, 199  
 add a `wchar_t` character (with attributes) to a curses window and advance cursor — `curs_addwch`  
     `wechowchar`, 194  
 add a character (with rendition) to a window  
     — `addch`, 39  
     — `mvaddch`, 39  
     — `mvwaddch`, 39  
     — `waddch`, 39  
 add a complex character (with rendition) to a window  
     — `add_wch`, 59  
     — `mvadd_wch`, 59  
     — `mvwadd_wch`, 59  
     — `wadd_wch`, 59  
 add a complex character and refresh window  
     — `echo_wchar`, 361  
     — `wecho_wchar`, 361  
 add a multi-byte character string (without rendition) to a window  
     — `addnstr`, 48  
     — `addstr`, 48  
     — `mvaddnstr`, 48  
     — `mvaddstr`, 48  
     — `mvwaddstr`, 48  
     — `mwwaddnstr`, 48  
     — `waddnstr`, 48  
     — `waddstr`, 48  
 add a single-byte border to a window  
     — `border`, 113  
     — `box`, 113  
     — `wborder`, 113  
 add a single-byte character and refresh window  
     — `echochar`, 357  
     — `wechochar`, 357  
 add a wide-character string to a window  
     — `addnwstr`, 51  
     — `addwstr`, 51  
     — `mvaddnwstr`, 51  
     — `mvaddwstr`, 51  
     — `mvwaddnwstr`, 51  
     — `mvwaddwstr`, 51  
     — `waddnwstr`, 51  
     — `waddwstr`, 51  
 add character and refresh window  
     — `pecho_wchar`, 1098  
     — `pechochar`, 1098  
`add_wch` — add a complex character (with rendition) to a window, 59  
`add_wchnstr` — copy a string of complex characters (with renditions) to a window, 63  
`add_wchstr` — copy a string of complex characters (with renditions) to a window, 63  
`addch` — add a character (with rendition) to a window, 39  
`addchnstr` — copy a character string (with renditions) to a window, 45  
`addchstr` — copy a character string (with renditions) to a window, 45  
`addnstr` — add a multi-byte character string (without rendition) to a window, 48  
`addnwstr` — add a string of `wchar_t` characters to a curses window and advance cursor, 199  
`addnwstr` — add a wide-character string to a window, 51  
`addstr` — add a multi-byte character string (without rendition) to a window, 48  
`addwch` — add a `wchar_t` character (with attributes) to a curses window and advance cursor, 194  
`addwchnstr` — add string of `wchar_t` characters (and attributes) to a curses window, 197  
`addwchstr` — add string of `wchar_t` characters (and attributes) to a curses window, 197  
`addwstr` — add a string of `wchar_t` characters to a curses window and advance cursor, 199  
`addwstr` — add a wide-character string to a window, 51  
`adjcurspos` — moving the cursor by character, 200

ALE curses library  
*See* curses library  
 apply padding information and output string  
   — `putp`, 1126  
   — `tputs`, 1126  
 arc — graphics interface, 1102  
 attr\_get — control window attributes, 76  
 attr\_off — control window attributes, 76  
 attr\_on — control window attributes, 76  
 attr\_set — control window attributes, 76  
 attroff — change foreground window attributes, 82  
 attroff — curses character and window attribute control routines, 201  
 attron — change foreground window attributes, 82  
 attron — curses character and window attribute control routines, 201  
 attrset — change foreground window attributes, 82  
 attrset — curses character and window attribute control routines, 201

## B

baudrate — return terminal baud rate, 98  
 beep — activate audio-visual alarm, 100  
 bkgd — set or get the background character (and rendition) of window, 102  
 bkgdset — set or get the background character (and rendition) of window, 102  
 bkgrnd — set or get the background character (and rendition) of window using a complex character, 107  
 bkgrndset — set or get the background character (and rendition) of window using a complex character, 107  
 border — add a single-byte border to a window, 113  
 border\_set — use complex characters (and renditions) to draw borders, 115  
 box — add a single-byte border to a window, 113  
 box — graphics interface, 1102  
 box\_set — use complex characters (and renditions) to draw borders, 115

## C

call refresh on changes to window — `immedok`, 529  
 can\_change\_color — manipulate color information, 132  
 cbreak — set input mode controls, 138  
 change foreground window attributes  
   — `attroff`, 82  
   — `attron`, 82  
   — `attrset`, 82  
   — `wattroff`, 82  
   — `wattron`, 82  
   — `wattrset`, 82  
 change the rendition of characters in a window  
   — `chgat`, 139  
   — `mvchgat`, 139  
   — `mvwchgat`, 139  
   — `wchgat`, 139  
 character based forms package, — `forms`, 435  
 character based menus package, — `menus`, 709  
 character based panels package, — `panels`, 1089  
 check for type-ahead characters — `typeahead`, 1460  
 chgat — change the rendition of characters in a window, 139  
 circle — graphics interface, 1102  
 clear — clear a window, 145  
 clear a window  
   — `clear`, 145  
   — `erase`, 145  
   — `wclear`, 145  
   — `werase`, 145  
 clear to the end of a line  
   — `clrtoeol`, 160  
   — `wclrtoeol`, 160  
 clear to the end of a window  
   — `clrrobot`, 158  
   — `wclrrobot`, 158  
 clearok — set terminal output controls, 149  
 closepl — graphics interface, 1102  
 closevt — graphics interface, 1102  
 clrrobot — clear to the end of a window, 158  
 clrtoeol — clear to the end of a line, 160  
 color\_content — manipulate color information, 132  
 COLOR\_PAIR — manipulate color information, 132

COLOR\_PAIRS — manipulate color information, 132

color\_set — control window attributes, 76

COLORS — manipulate color information, 132

COLS — number of columns on terminal screen, 178

cont — graphics interface, 1102

control flush of input and output on interrupt

- noqiflush, 1049
- qiflush, 1049

control window attributes

- attr\_get, 76
- attr\_off, 76
- attr\_on, 76
- attr\_set, 76
- color\_set, 76
- wattr\_get, 76
- wattr\_off, 76
- wattr\_on, 76
- wattr\_set, 76
- wcolor\_set, 76

control window refresh

- is\_linetouched, 608
- is\_wintouched, 608
- touchline, 608
- touchwin, 608
- untouchwin, 608
- wtouchln, 608

convert formatted input from a window — mvscanw, 864

convert formatted input from a window — mvwscanw, 864

convert formatted input from a window — scanw, 864

convert formatted input from a window — wscanw, 864

convert formatted input from a window — vwscanw, 1511

copy overlapped windows — overlay, 1066

copy overlapped windows — overwrite, 1066

copy a character string (with renditions) to a window

- addchnstr, 45
- addchstr, 45
- mvaddchnstr, 45
- mvaddchstr, 45
- mvwaddchnstr, 45
- mvwaddchstr, 45

copy a character string (with renditions) to a window (Continued)

- waddchnstr, 45
- waddchstr, 45

copy a string of complex characters (with renditions) to a window

- add\_wchnstr, 63
- add\_wchstr, 63
- mvadd\_wchnstr, 63
- mvadd\_wchstr, 63
- mvwadd\_wchnstr, 63
- mvwadd\_wchstr, 63
- wadd\_wchnstr, 63
- wadd\_wchstr, 63

copywin — overlay or overwrite any portion of window, 183

create a new window or subwindow

- derwin, 339
- newwin, 339
- subwin, 339

create or refresh a pad or subpad

- newpad, 1013
- pnoutrefresh, 1013
- prefresh, 1013
- subpad, 1013

CRT handling and optimization package, — curses, 214

cur\_term — current terminal information, 306

current terminal information — cur\_term, 306

current window — curscr, 211

cur\_addwch — add a wchar\_t character (with attributes) to a curses window and advance cursor, 194

cur\_addwchstr — add string of wchar\_t characters (and attributes) to a curses window, 197

cur\_addwstr — add a string of wchar\_t characters to a curses window and advance cursor, 199

cur\_alecompat — moving the cursor by character, 200

cur\_attr — curses character and window attribute control routines, 201

- Attributes, 201

cur\_getwch — get (or push back) wchar\_t characters from curses terminal keyboard, 246

- Function Keys, 246

curs\_getwstr — get wchar\_t character strings from curses terminal keyboard, 251  
 curs\_inswch — insert a wchar\_t character before the character under the cursor in a curses window, 264  
 curs\_inswstr — insert wchar\_t string before character under the cursor in a curses window, 265  
 curs\_inwch — get a wchar\_t character and its attributes from a curses window, 267  
 curs\_inwchstr — get a string of wchar\_t characters (and attributes) from a curses window, 268  
 curs\_inwstr — get a string of wchar\_t characters from a curses window, 269  
 curs\_pad — create and display curses pads, 277  
 curs\_set — set visibility of cursor, 288  
 curscr — current window, 211  
 curses — CRT handling and optimization package, 214  
 curses — introduction and overview of X/Open Curses, 229  
 curses pads, create and display — curs\_pad, 277  
 curses — introduction and overview of X/Open Curses  
   Attributes, Color Pairs, and Renditions, 231  
   Complex Characters, 234  
   Data Types, 230  
   Display Operations, 234  
   Input Processing, 236  
 curses pads, create and display — curs\_pad  
   newpad, 277  
 curses — introduction and overview of X/Open Curses  
   Non-Spacing Characters, 233  
   Overlapping Windows, 235  
 curses pads, create and display — curs\_pad  
   pechochar, 277  
   pechowchar, 277  
   pnoutrefresh, 277  
   prefresh, 277  
 curses — introduction and overview of X/Open Curses  
   Screens, Windows, and Terminals, 230  
   Special Characters, 235  
 curses pads, create and display — curs\_pad  
   subpad, 277  
 curses, low-level routines  
   — curs\_kernel, 270  
   — curs\_set, 270  
   — def\_prog\_mode, 270  
   — def\_shell\_mode, 270  
   — getsyx, 270  
   — napms, 270  
   — reset\_prog\_mode, 270  
   — reset\_shell\_mode, 270  
   — resettty, 270  
   — ripoffline, 270  
   — savetty, 270  
   — setsyx, 270  
 curses bell and screen flash routines  
   — beep, 203  
   — curs\_beep, 203  
   — flash, 203  
 curses borders, horizontal and vertical lines, create  
   — border, 205  
   — box, 205  
   — curs\_border, 205  
   — wborder, 205  
   — whline, 205  
   — wvline, 205  
 curses character and window attribute control routines  
   — attroff, 201  
   — attron, 201  
   — attrset, 201  
   — curs\_attr, 201  
   — standend, 201  
   — standout, 201  
   — wattroff, 201  
   — wattron, 201  
   — wattrset, 201  
   — wstandend, 201  
   — wstandout, 201  
 curses color manipulation routines  
   — can\_change\_colors, 208  
   — color\_content, 208  
   — curs\_color, 208  
   — has\_colors, 208  
   — init\_color, 208  
   — init\_pair, 208  
   — pair\_content, 208

curses color manipulation routines (Continued)

— start\_color, 208

curses cursor and window coordinates

— curs\_getyx, 252

— getbegyx, 252

— getmaxyx, 252

— getparyx, 252

— getyx, 252

curses environment query routines

— baudrate, 291

— curs\_termattrs, 291

— erasechar, 291

— has\_ic, 291

— has\_il, 291

— killchar, 291

— longname, 291

— termattrs, 291

— termname, 291

curses interfaces to termcap library

— curs\_termcap, 293

— tgetent, 293

— tgetflag, 293

— tgetnum, 293

— tgetstr, 293

— tgoto, 293

— tputs, 293

curses interfaces to terminfo database

— curs\_terminfo, 295

— del\_curterm, 295

— mvcur, 295

— putp, 295

— restartterm, 295

— set\_curterm, 295

— setterm, 295

— setupterm, 295

— tigetflag, 295

— tigetnum, 295

— tigetstr, 295

— tparm, 295

— tputs, 295

— vidattr, 295

— vidputs, 295

curses library

*See also* form library, menu library, or panel library

— adjcurspos, 200

— curs\_alecompat, 200

— movenextch, 200

curses library (Continued)

— moveprevch, 200

— wadjcurspos, 200

— wmovenextch, 200

— wmoveprevch, 200

curses miscellaneous utility routines

— curs\_util, 301

— delay\_output, 301

— filter, 301

— flushinp, 301

— getwin, 301

— keyname, 301

— putwin, 301

— unctrl, 301

— use\_env, 301

curses refresh control routines

— curs\_touch, 299

— is\_linetouched, 299

— is\_wintouched, 299

— touchline, 299

— touchwin, 299

— untouchwin, 299

— wtouchln, 299

curses screen, read/write from/to file

— curs\_scr\_dump, 283

— scr\_dump, 283

— scr\_init, 283

— scr\_restore, 283

— scr\_set, 283

curses screen initialization and manipulation routines

— curs\_initscr, 255

— delscreen, 255

— endwin, 255

— initscr, 255

— isendwin, 255

— newterm, 255

— set\_term, 255

curses soft label routines

— curs\_slk, 289

— slk\_attroff, 289

— slk\_attron, 289

— slk\_attrset, 289

— slk\_clear, 289

— slk\_init, 289

— slk\_label, 289

— slk\_noutrefresh, 289

— slk\_refresh, 289

curses soft label routines (Continued)

- slk\_restore, 289
- slk\_set, 289
- slk\_touch, 289

curses terminal input option control routines

- cbreak, 257
- curs\_inopts, 257
- echo, 257
- halfdelay, 257
- intrflush, 257
- keypad, 257
- meta, 257
- nocbreak, 257
- nodelay, 257
- noecho, 257
- noqiflush, 257
- noraw, 257
- notimeout, 257
- qiflush, 257
- raw, 257
- timeout, 257
- typeahead, 257
- wtimeout, 257

curses terminal keyboard

- curs\_getstr, 245
- getstr, 245
- mvgetstr, 245
- mvwgetstr, 245
- wgetnstr, 245
- wgetstr, 245

curses terminal keyboard, get characters

- curs\_getch, 240
- getch, 240
- mvgetch, 240
- mvwgetch, 240
- ungetch, 240
- wgetch, 240

curses terminal output option control routines

- clearok, 273
- curs\_outopts, 273
- idcok, 273
- idlok, 273
- immedok, 273
- leaveok, 273
- nl, 273
- nonl, 273
- scrollok, 273
- setscreg, 273

curses terminal output option control routines (Continued)

- wsetscreg, 273

curses window, add character and advance

- cursor
- addch, 189
- curs\_addch, 189
- echochar, 189
- mvwaddch, 189
- waddch, 189
- wechochar, 189

curses window, add string of characters

- addchnstr, 192
- addchstr, 192
- curs\_addchstr, 192
- mvaddchnstr, 192
- mvaddchstr, 192
- mvwaddchnstr, 192
- mvwaddchstr, 192
- waddchnstr, 192
- waddchstr, 192

curses window, add string of characters and advance cursor

- addnstr, 193
- addstr, 193
- curs\_addstr, 193
- mvaddnstr, 193
- mvaddstr, 193
- mvwaddstr, 193
- waddnstr, 193
- waddstr, 193

curses window, clear all or part

- clear, 207
- clrrobot, 207
- clrtoeol, 207
- curs\_clear, 207
- erase, 207
- wclear, 207
- wclrrobot, 207
- wclrtoeol, 207
- werase, 207

curses window, convert formatted input

- curs\_scanw, 282
- mvscanw, 282
- mvwscanw, 282
- scanw, 282
- vwscanw, 282
- wscanw, 282

curses window, delete and insert lines

- `curs_deleteln`, 213
- `deleteln`, 213
- `insdelln`, 213
- `insertln`, 213
- `wdeleteln`, 213
- `winsdelln`, 213
- `winsertln`, 213

curses window, delete character under cursor

- `curs_delch`, 212
- `delch`, 212
- `mvdelch`, 212
- `mvwdelch`, 212
- `wdelch`, 212

curses window, get character and its attributes

- `curs_inch`, 253
- `inch`, 253
- `mvinch`, 253
- `mvwinch`, 253
- `winch`, 253

curses window, get string of characters

- `curs_inchstr`, 254
- `curs_instr`, 263
- `inchstr`, 254
- `inchstr`, 254
- `innstr`, 263
- `instr`, 263
- `mvinchnstr`, 254
- `mvinchstr`, 254
- `mvinnstr`, 263
- `mvinstr`, 263
- `mvwinchnstr`, 254
- `mvwinchstr`, 254
- `mvwinnstr`, 263
- `mvwinstr`, 263
- `winchnstr`, 254
- `winchstr`, 254
- `winnstr`, 263
- `winstr`, 263

curses window, insert character before character under cursor

- `curs_insch`, 260
- `insch`, 260
- `mvinsch`, 260
- `mvwinsch`, 260
- `winsch`, 260

curses window, insert string before character under cursor

- `curs_instr`, 261
- `insnstr`, 261
- `instr`, 261
- `mvinsnstr`, 261
- `mvinsstr`, 261
- `mvwinsnstr`, 261
- `mvwinsstr`, 261
- `winsnstr`, 261
- `winsstr`, 261

curses window, scroll

- `curs_scroll`, 285
- `sclr`, 285
- `scroll`, 285
- `wscrl`, 285

curses window background manipulation routines

- `bkgd`, 204
- `bkgdset`, 204
- `curs_bkgd`, 204
- `wbkgd`, 204
- `wbkgdset`, 204

curses window cursor

- `curs_move`, 272
- `move`, 272
- `wmove`, 272

curses windows, create

- `curs_window`, 303
- `delwin`, 303
- `derwin`, 303
- `dupwin`, 303
- `mvderwin`, 303
- `mvwin`, 303
- `newwin`, 303
- `subwin`, 303
- `syncok`, 303
- `wcursyncup`, 303
- `wsyncdown`, 303
- `wsyncup`, 303

curses windows, overlap and manipulate

- `copywin`, 276
- `curs_overlay`, 276
- `overlay`, 276
- `overwrite`, 276

curses windows, print formatted output

- `curs_printw`, 279
- `mvprintw`, 279

curses windows, print formatted output  
(Continued)

- mvwprintw, 279
- printw, 279
- vwprintw, 279
- wprintw, 279

curses windows and lines, refresh

- curs\_refresh, 280
- douupdate, 280
- redrawwin, 280
- refresh, 280
- wnoutrefresh, 280
- wredrawln, 280
- wrefresh, 280

## D

- def\_prog\_mode — save/restore terminal modes, 311
- def\_shell\_mode — save/restore terminal modes, 311
- default window — stdscr, 1370
- del\_curterm — interfaces to the terminfo database, 324
- delay\_output — delays output, 317
- delays output — delay\_output, 317
- delch — remove a character, 319
- delete a window — delwin, 335
- deleteln — remove a line, 327
- delwin — delete a window, 335
- derwin — create a new window or subwindow, 339
- determine insert/delete character/line capability
  - has\_ic, 508
  - has\_il, 508
- disable use of certain terminal capabilities — filter, 399
- discard type-ahead characters — flushinp, 404
- douupdate — refresh windows and lines, 343
- duplicate a window — dupwin, 348
- dupwin — duplicate a window, 348

## E

- echo — enable/disable terminal echo, 353

- echo\_wchar — add a complex character and refresh window, 361
- echochar — add a single-byte character and refresh window, 357
- echowchar — add a wchar\_t character (with attributes) to a curses window and advance cursor, 194
- emulate the termcap database
  - tgetent, 1393
  - tgetflag, 1393
  - tgetnum, 1393
  - tgetstr, 1393
  - tgoto, 1393
- enable/disable half-delay mode — halfdelay, 499
- enable/disable hardware insert-character and delete-character features — idcok, 520
- enable/disable keypad handling — keypad, 638
- enable/disable meta keys — meta, 721
- enable/disable newline control
  - nl, 1029
  - nonl, 1029
- enable/disable terminal echo
  - echo, 353
  - noecho, 353
- endwin — restore initial terminal environment, 364
- erase — clear a window, 145
- erase — graphics interface, 1102
- erasechar — return current ERASE or KILL characters, 372
- erasewchar — return current ERASE or KILL characters, 372

## F

- filter — disable use of certain terminal capabilities, 399
- flash — activate audio-visual alarm, 100
- enable or disable flush on interrupt — intrflush, 591
- flushinp — discard type-ahead characters, 404
- form library
  - See also* curses library
- forms — character based forms package, 435

- forms, application-specific routines
  - field\_init, 423
  - field\_term, 423
  - form\_hook, 423
  - form\_init, 423
  - form\_term, 423
  - set\_field\_init, 423
  - set\_field\_term, 423
  - set\_form\_init, 423
  - set\_form\_term, 423
- forms, associate application data
  - field\_userptr, 421
  - form\_field\_userptr, 421
  - form\_userptr, 442
  - set\_field\_userptr, 421
  - set\_form\_userptr, 442
- forms, command processor, — form\_driver, 407
- forms, connect fields
  - field\_count, 410
  - form\_field, 410
  - form\_fields, 410
  - move\_field, 410
  - set\_form\_fields, 410
- forms, create and destroy
  - form\_new, 427
  - free\_form, 427
  - new\_form, 427
- forms, format general appearance
  - field\_just, 414
  - form\_field\_just, 414
  - set\_field\_just, 414
- forms, format general display attributes
  - field\_back, 411
  - field\_fore, 411
  - field\_pad, 411
  - form\_field\_attributes, 411
  - set\_field\_back, 411
  - set\_field\_fore, 411
  - set\_field\_pad, 411
- forms, set current page and field
  - current\_field, 432
  - field\_index, 432
  - form\_page, 432
  - set\_current\_field, 432
  - set\_form\_page, 432
- forms, write/erase from associated subwindows
  - form\_post, 434
- forms, write/erase from associated subwindows (Continued)
  - post\_form, 434
  - unpost\_form, 434
- forms field, off-screen data ahead or behind
  - data\_ahead, 406
  - data\_behind, 406
  - form\_data, 406
- forms field attributes, set and get
  - field\_buffer, 412
  - field\_status, 412
  - form\_field\_buffer, 412
  - set\_field\_buffer, 412
  - set\_field\_status, 412
  - set\_max\_field, 412
- forms field characteristics
  - dynamic\_field\_info, 413
  - field\_info, 413
  - form\_field\_info, 413
- forms field data type validation
  - field\_arg, 422
  - field\_type, 422
  - form\_field\_validation, 422
  - set\_field\_type, 422
- forms field option routines
  - field\_opts, 416
  - field\_opts\_off, 416
  - field\_opts\_on, 416
  - form\_field\_opts, 416
  - set\_field\_opts, 416
- forms fields, create and destroy
  - dup\_field, 415
  - form\_field\_new, 415
  - free\_field, 415
  - link\_field, 415
  - new\_field, 415
- forms fieldtype routines
  - form\_fieldtype, 419
  - free\_fieldtype, 419
  - link\_fieldtype, 419
  - new\_fieldtype, 419
  - set\_fieldtype\_arg, 419
  - set\_fieldtype\_choice, 419
- forms option routines
  - form\_opts, 429
  - form\_opts\_off, 429
  - form\_opts\_on, 429
  - set\_form\_opts, 429

- forms pagination
  - `form_new_page`, 428
  - `new_page`, 428
  - `set_new_page`, 428
- forms window and subwindow association routines
  - `form_sub`, 443
  - `form_win`, 443
  - `scale_form`, 443
  - `set_form_sub`, 443
  - `set_form_win`, 443
- forms window cursor, position
  - `form_cursor`, 405
  - `pos_form_cursor`, 405

## G

- generate printable representation of a character
  - `unctrl`, 1463
- generate printable representation of a wide character
  - `wunctrl`, 1751
- get a string of `wchar_t` characters (and attributes) from a curses window
  - `curs_inwchstr`, 268
- get a string of `wchar_t` characters from a curses window
  - `curs_inwstr`, 269
- get a `wchar_t` character and its attributes from a curses window
  - `curs_inwch`, 267
- get supported terminal video attributes
  - `termattrs`, 1387
- get supported terminal video attributes
  - `term_attrs`, 1387
- get `wchar_t` character strings from curses terminal keyboard
  - `curs_getwstr`, 251
  - `getnwstr`, 251
- get `wchar_t` character strings from curses terminal keyboard
  - `curs_getwstr`, `getwstr`, 251
- get a string of `wchar_t` characters from a curses window
  - `curs_inwstr`
  - `innwstr`, 269
- get a `wchar_t` character and its attributes from a curses window
  - `curs_inwch`
  - `inwch`, 267
- get a string of `wchar_t` characters (and attributes) from a curses window
  - `curs_inwchstr`

- get a string of `wchar_t` characters (and attributes) from a curses window
  - `curs_inwchstr` (Continued)
  - `inwchnstr`, 268
  - `inwchstr`, 268
- get a string of `wchar_t` characters from a curses window
  - `curs_inwstr`
  - `inwstr`, 269
- get `wchar_t` character strings from curses terminal keyboard
  - `curs_getwstr`
  - `mvgetnwstr`, 251
  - `mvgetwstr`, 251
- get a string of `wchar_t` characters from a curses window
  - `curs_inwstr`
  - `mvinnwstr`, 269
- get a `wchar_t` character and its attributes from a curses window
  - `curs_inwch`
  - `mvinwch`, 267
- get a string of `wchar_t` characters (and attributes) from a curses window
  - `curs_inwchstr`
  - `mvinwchnstr`, 268
  - `mvinwchstr`, 268
- get a string of `wchar_t` characters from a curses window
  - `curs_inwstr`
  - `mvinwstr`, 269
- get `wchar_t` character strings from curses terminal keyboard
  - `curs_getwstr`
  - `mvwgetnwstr`, 251
  - `mvwgetwstr`, 251
- get a string of `wchar_t` characters from a curses window
  - `curs_inwstr`
  - `mvwinnwstr`, 269
- get a `wchar_t` character and its attributes from a curses window
  - `curs_inwch`
  - `mvwinwch`, 267
- get a string of `wchar_t` characters (and attributes) from a curses window
  - `curs_inwchstr`
  - `mvwinwchnstr`, 268
  - `mvwinwchstr`, 268
- get a string of `wchar_t` characters from a curses window
  - `curs_inwstr`
  - `mvwinwstr`, 269
- get `wchar_t` character strings from curses terminal keyboard
  - `curs_getwstr`
  - `wgetnwstr`, 251
  - `wgetwstr`, 251

get a string of `wchar_t` characters from a curses window — `curs_inwstr`  
`winnwstr`, 269

get a `wchar_t` character and its attributes from a curses window — `curs_inwch`  
`winwch`, 267

get a string of `wchar_t` characters (and attributes) from a curses window — `curs_inwchstr`  
`winwchnstr`, 268  
`winwchstr`, 268

get a string of `wchar_t` characters from a curses window — `curs_inwstr`  
`winwstr`, 269

get (or push back) `wchar_t` characters from curses terminal keyboard  
 — `curs_getwch`, 246  
 — `getwch`, 246  
 — `mvgetwch`, 246  
 — `mvwgetwch`, 246  
 — `ungetwch`, 246  
 — `wgetwch`, 246

get a multibyte character string from terminal  
 — `getnstr`, 469  
 — `getstr`, 469  
 — `mvgetnstr`, 469  
 — `mvgetstr`, 469  
 — `mvwgetnstr`, 469  
 — `mvwgetstr`, 469  
 — `wgetnstr`, 469  
 — `wgetstr`, 469

get a single-byte character from terminal  
 — `getch`, 462  
 — `mvgetch`, 462  
 — `mvwgetch`, 462  
 — `wgetch`, 462

get a wide character from terminal  
 — `get_wch`, 486  
 — `mvget_wch`, 486  
 — `mvwget_wch`, 486  
 — `wget_wch`, 486

get a wide character string (with rendition) from a `cchar_t` — `getcchar`, 456

get a wide character string from terminal  
 — `get_wstr`, 472  
 — `getn_wstr`, 472  
 — `mvget_wstr`, 472  
 — `mvgetn_wstr`, 472

get a wide character string from terminal  
 (Continued)  
 — `mvwget_wstr`, 472  
 — `mvwgetn_wstr`, 472  
 — `wget_wstr`, 472  
 — `wgetn_wstr`, 472

get cursor or window coordinates  
 — `getbegyx`, 451  
 — `getmaxyx`, 451  
 — `getparyx`, 451  
 — `getyx`, 451

`get_wch` — get a wide character from terminal, 486

`get_wstr` — get a wide character string from terminal, 472

`getbegyx` — get cursor or window coordinates, 451

`getbkgd` — set or get the background character (and rendition) of window, 102

`getbkgrnd` — set or get the background character (and rendition) of window using a complex character, 107

`getcchar` — get a wide character string (with rendition) from a `cchar_t`, 456

`getch` — get a single-byte character from terminal, 462

`getmaxyx` — get cursor or window coordinates, 451

`getn_wstr` — get a wide character string from terminal, 472

`getnstr` — get a multibyte character string from terminal, 469

`getnwstr` — get `wchar_t` character strings from curses terminal keyboard, 251

`getparyx` — get cursor or window coordinates, 451

`getstr` — get a multibyte character string from terminal, 469

`getwch` — get (or push back) `wchar_t` characters from curses terminal keyboard, 246

`getwin` — read a window from, and write a window to, a file, 490

`getwstr` — get `wchar_t` character strings from curses terminal keyboard, 251

`getyx` — get cursor or window coordinates, 451  
 graphics interface  
 — `arc`, 1102  
 — `box`, 1102

graphics interface (Continued)

- circle, 1102
- closepl, 1102
- closevt, 1102
- cont, 1102
- erase, 1102
- label, 1102
- line, 1102
- linemod, 1102
- move, 1102
- openpl, 1102
- openvt, 1102
- plot, 1102
- point, 1102
- space, 1102

**H**

- halfdelay — enable/disable half-delay mode, 499
- has\_colors — manipulate color information, 132
- has\_ic — determine insert/delete character/line capability, 508
- has\_il — determine insert/delete character/line capability, 508
- hline — use single-byte characters (and renditions) to draw lines, 513
- hline\_set — use complex characters (and renditions) to draw lines, 515

**I**

- idcok — enable/disable hardware insert-character and delete-character features, 520
- idlok — set terminal output controls, 149
- immedok — call refresh on changes to window, 529
- in\_wch — retrieve a complex character (with rendition), 593
- in\_wchnstr — retrieve complex character string (with rendition), 595
- in\_wchstr — retrieve complex character string (with rendition), 595

- inch — return a single-byte character (with rendition), 531
- inchnstr — retrieve a single-byte character string (with rendition), 533
- inchstr — retrieve a single-byte character string (with rendition), 533
- init\_color — manipulate color information, 132
- init\_pair — manipulate color information, 132
- initscr — screen initialization functions, 552
- innstr — retrieve a multibyte character string (without rendition), 555
- innwstr — get a string of wchar\_t characters from a curses window, 269
- innwstr — retrieve a wide character string (without rendition), 558
- ins\_nwstr — insert a wide character string, 572
- ins\_wch — insert a complex character, 582
- ins\_wstr — insert a wide character string, 572
- insch — insert a character, 561
- insdelln — insert/delete lines to/from the window, 563
- insert a wchar\_t character before the character under the cursor in a curses window — curs\_inswch, 264
- insert wchar\_t string before character under the cursor in a curses window — curs\_inswstr, 265
- insnwstr, 265
- insert a wchar\_t character before the character under the cursor in a curses window — curs\_inswch
- inswch, 264
- insert wchar\_t string before character under the cursor in a curses window — curs\_inswstr
- inswstr, 265
- mvinsnwstr, 265
- insert a wchar\_t character before the character under the cursor in a curses window — curs\_inswch
- mvinswch, 264
- insert wchar\_t string before character under the cursor in a curses window — curs\_inswstr
- mvinswstr, 265
- mvwinsnwstr, 265
- insert a wchar\_t character before the character under the cursor in a curses window — curs\_inswch
- mvwinswch, 264

insert `wchar_t` string before character under the cursor in a curses window —  `curs_inswstr`,  `mvwinswstr`, 265  
 `winswstr`, 265  
 insert a `wchar_t` character before the character under the cursor in a curses window —  `curs_inswch`,  `winswch`, 264  
 insert `wchar_t` string before character under the cursor in a curses window —  `curs_inswstr`,  `winswstr`, 265  
 insert a character  
   —  `insch`, 561  
   —  `mvinsch`, 561  
   —  `mvwinsch`, 561  
   —  `winsch`, 561  
 insert a complex character  
   —  `ins_wch`, 582  
   —  `mvins_wch`, 582  
   —  `mvwins_wch`, 582  
   —  `wins_wch`, 582  
 insert a line in a window  
   —  `insertln`, 565  
   —  `winsertln`, 565  
 insert a multibyte character string  
   —  `insstr`, 568  
   —  `insstr`, 568  
   —  `mvinsnstr`, 568  
   —  `mvinsstr`, 568  
   —  `mvwinsnstr`, 568  
   —  `mvwinsstr`, 568  
   —  `winsnstr`, 568  
   —  `winsstr`, 568  
 insert a wide character string  
   —  `ins_nwstr`, 572  
   —  `ins_wstr`, 572  
   —  `mvins_nwstr`, 572  
   —  `mvins_wstr`, 572  
   —  `mvwins_nwstr`, 572  
   —  `mvwins_wstr`, 572  
   —  `wins_nwstr`, 572  
   —  `wins_wstr`, 572  
 insert/delete lines to/from the window  
   —  `insdelln`, 563  
   —  `windselln`, 563  
 `insertln` — insert a line in a window, 565  
 `insstr` — insert a multibyte character string, 568  
 `insnwstr` — insert `wchar_t` string before character under the cursor in a curses window, 265  
 `insstr` — insert a multibyte character string, 568  
 `instr` — retrieve a multibyte character string (without rendition), 555  
 `inswch` — insert a `wchar_t` character before the character under the cursor in a curses window, 264  
 `inswstr` — insert `wchar_t` string before character under the cursor in a curses window, 265  
 interfaces to the terminfo database —  `del_curterm`, 324  
 interfaces to the terminfo database —  `restartterm`, 324  
 interfaces to the terminfo database —  `setupterm`, 324  
 interfaces to the terminfo database —  `set_curterm`, 324  
 `intrflush` — enable or disable flush on interrupt, 591  
 introduction and overview of X/Open Curses —  `curses`, 229  
 `inwch` — get a `wchar_t` character and its attributes from a curses window, 267  
 `inwchnstr` — get a string of `wchar_t` characters (and attributes) from a curses window, 268  
 `inwchstr` — get a string of `wchar_t` characters (and attributes) from a curses window, 268  
 `inwstr` — get a string of `wchar_t` characters from a curses window, 269  
 `inwstr` — retrieve a wide character string (without rendition), 558  
 `is_linetouched` — control window refresh, 608  
 `is_wintouched` — control window refresh, 608  
 `isendwin` — restore initial terminal environment, 364

## K

`key_name` — return character string used as key name, 633  
 `keyname` — return character string used as key name, 633  
 `keypad` — enable/disable keypad handling, 638

killchar — return current ERASE or KILL characters, 372  
killwchar — return current ERASE or KILL characters, 372

## L

label — graphics interface, 1102  
leaveok — set terminal output controls, 149  
line — graphics interface, 1102  
linemod — graphics interface, 1102  
LINES — number of lines on terminal screen, 668  
longname — return full terminal type name, 674

## M

manipulate color information —  
  can\_change\_color, 132  
manipulate color information — COLORS, 132  
manipulate color information —  
  color\_content, 132  
manipulate color information —  
  COLOR\_PAIR, 132  
manipulate color information —  
  COLOR\_PAIRS, 132  
manipulate color information —  
  has\_colors, 132  
manipulate color information — init\_color, 132  
manipulate color information — init\_pair, 132  
manipulate color information —  
  pair\_content, 132  
manipulate color information —  
  PAIR\_NUMBER, 132  
manipulate color information —  
  start\_color, 132  
map area of parent window to subwindow —  
  mvderwin, 779  
menu library  
  *See also* curses library  
menus — character based menus package, 709  
menus, application-specific routines  
  — item\_init, 687  
  — item\_term, 687  
  — menu\_hook, 687

menus, application-specific routines  
(Continued)

  — menu\_init, 687  
  — menu\_term, 687  
  — set\_item\_init, 687  
  — set\_item\_term, 687  
  — set\_menu\_init, 687  
  — set\_menu\_term, 687  
menus, associate application data  
  — menu\_userptr, 716  
  — set\_menu\_userptr, 716  
menus, create and destroy  
  — free\_menu, 701  
  — menu\_new, 701  
  — new\_menu, 701  
menus, rows and columns  
  — menu\_format, 684  
  — set\_menu\_format, 684  
menus cursor  
  — menu\_cursor, 679  
  — pos\_menu\_cursor, 679  
menus display attributes  
  — menu\_attributes, 675  
  — menu\_back, 675  
  — menu\_fore, 675  
  — menu\_grey, 675  
  — menu\_pad, 675  
  — set\_menu\_back, 675  
  — set\_menu\_fore, 675  
  — set\_menu\_grey, 675  
  — set\_menu\_pad, 675  
menus from associated subwindows,  
  write/erase  
  — menu\_post, 708  
  — post\_menu, 708  
  — unpost\_menu, 708  
menus item, visibility  
  — item\_visible, 699  
  — menu\_item\_visible, 699  
menus item name and description  
  — item\_description, 693  
  — item\_name, 693  
  — menu\_item\_name, 693  
menus item options routines  
  — item\_opts, 695  
  — item\_opts\_off, 695  
  — item\_opts\_on, 695  
  — menu\_item\_opts, 695

menus item options routines (Continued)  
   — set\_item\_opts, 695  
 menus item values, set and get  
   — item\_value, 698  
   — menu\_item\_value, 698  
   — set\_item\_value, 698  
 menus items, associate application data  
   — item\_userptr, 697  
   — menu\_item\_userptr, 697  
   — set\_item\_userptr, 697  
 menus items, connect and disconnect  
   — item\_count, 696  
   — menu\_items, 696  
   — set\_menu\_items, 696  
 menus items, create and destroy  
   — free\_item, 694  
   — menu\_item\_new, 694  
   — new\_item, 694  
 menus items, get and set  
   — current\_item, 691  
   — item\_index, 691  
   — menu\_item\_current, 691  
   — set\_current\_item, 691  
   — set\_top\_row, 691  
   — top\_row, 691  
 menus mark string routines  
   — menu\_mark, 700  
   — set\_menu\_mark, 700  
 menus options routines  
   — menu\_opts, 702  
   — menu\_opts\_off, 702  
   — menu\_opts\_on, 702  
   — set\_menu\_opts, 702  
 menus pattern match buffer  
   — menu\_pattern, 707  
   — set\_menu\_pattern, 707  
 menus subsystem, command processor, —  
   menu\_driver, 680  
 menus window and subwindow association  
   routines  
   — menu\_sub, 717  
   — menu\_win, 717  
   — scale\_menu, 717  
   — set\_menu\_sub, 717  
   — set\_menu\_win, 717  
 meta — enable/disable meta keys, 721  
 move — graphics interface, 1102  
 move — move cursor in window, 726  
   — move, 726  
   — wmove, 726  
 move the cursor — mvcur, 773  
 move window — mvwin, 944  
 movenextch — moving the cursor by  
   character, 200  
 moveprevch — moving the cursor by  
   character, 200  
 mvadd\_wch — add a complex character (with  
   rendition) to a window, 59  
 mvadd\_wchnstr — copy a string of complex  
   characters (with renditions) to a window, 63  
 mvadd\_wchstr — copy a string of complex  
   characters (with renditions) to a window, 63  
 mvaddch — add a character (with rendition) to  
   a window, 39  
 mvaddchnstr — copy a character string (with  
   renditions) to a window, 45  
 mvaddchstr — copy a character string (with  
   renditions) to a window, 45  
 mvaddnstr — add a multi-byte character string  
   (without rendition) to a window, 48  
 mvaddnwstr — add a string of wchar\_t  
   characters to a curses window and advance  
   cursor, 199  
 mvaddnwstr — add a wide-character string to a  
   window, 51  
 mvaddstr — add a multi-byte character string  
   (without rendition) to a window, 48  
 mvaddwch — add a wchar\_t character (with  
   attributes) to a curses window and advance  
   cursor, 194  
 mvaddwchnstr — add string of wchar\_t  
   characters (and attributes) to a curses  
   window, 197  
 mvaddwchstr — add string of wchar\_t  
   characters (and attributes) to a curses  
   window, 197  
 mvaddwstr — add a string of wchar\_t  
   characters to a curses window and advance  
   cursor, 199  
 mvaddwstr — add a wide-character string to a  
   window, 51  
 mvchgat — change the rendition of characters  
   in a window, 139  
 mvcur — move the cursor, 773  
 mvdelch — remove a character, 319

mvderwin — map area of parent window to subwindow, 779  
 mvget\_wch — get a wide character from terminal, 486  
 mvget\_wstr — get a wide character string from terminal, 472  
 mvgetch — get a single-byte character from terminal, 462  
 mvgetn\_wstr — get a wide character string from terminal, 472  
 mvgetnstr — get a multibyte character string from terminal, 469  
 mvgetnwstr — get wchar\_t character strings from curses terminal keyboard, 251  
 mvgetstr — get a multibyte character string from terminal, 469  
 mvgetwch — get (or push back) wchar\_t characters from curses terminal keyboard, 246  
 mvgetwstr — get wchar\_t character strings from curses terminal keyboard, 251  
 mvhline — use single-byte characters (and renditions) to draw lines, 513  
 mvhline\_set — use complex characters (and renditions) to draw lines, 515  
 mvin\_wch — retrieve a complex character (with rendition), 593  
 mvin\_wchnstr — retrieve complex character string (with rendition), 595  
 mvin\_wchstr — retrieve complex character string (with rendition), 595  
 mvinch — return a single-byte character (with rendition), 531  
 mvinchnstr — retrieve a single-byte character string (with rendition), 533  
 mvinchstr — retrieve a single-byte character string (with rendition), 533  
 mvinnstr — retrieve a multibyte character string (without rendition), 555  
 mvinnwstr — get a string of wchar\_t characters from a curses window, 269  
 mvinnwstr — retrieve a wide character string (without rendition), 558  
 mvins\_nwstr — insert a wide character string, 572  
 mvins\_wch — insert a complex character, 582  
 mvins\_wstr — insert a wide character string, 572  
 mvinsch — insert a character, 561  
 mvinsnstr — insert a multibyte character string, 568  
 mvinsnwstr — insert wchar\_t string before character under the cursor in a curses window, 265  
 mvinsstr — insert a multibyte character string, 568  
 mvinstr — retrieve a multibyte character string (without rendition), 555  
 mvinswch — insert a wchar\_t character before the character under the cursor in a curses window, 264  
 mvinswstr — insert wchar\_t string before character under the cursor in a curses window, 265  
 mvinwch — get a wchar\_t character and its attributes from a curses window, 267  
 mvinwchnstr — get a string of wchar\_t characters (and attributes) from a curses window, 268  
 mvinwchstr — get a string of wchar\_t characters (and attributes) from a curses window, 268  
 mvinwstr — get a string of wchar\_t characters from a curses window, 269  
 mvinwstr — retrieve a wide character string (without rendition), 558  
 mvprintw — print formatted output window, 862  
 mvscanw — convert formatted input from a window, 864  
 mvvline — use single-byte characters (and renditions) to draw lines, 513  
 mvvline\_set — use complex characters (and renditions) to draw lines, 515  
 mvwadd\_wch — add a complex character (with rendition) to a window, 59  
 mvwadd\_wchnstr — copy a string of complex characters (with renditions) to a window, 63  
 mvwadd\_wchstr — copy a string of complex characters (with renditions) to a window, 63  
 mvwaddch — add a character (with rendition) to a window, 39  
 mvwaddchnstr — copy a character string (with renditions) to a window, 45  
 mvwaddchstr — copy a character string (with renditions) to a window, 45

mvwaddnwstr — add a string of wchar\_t characters to a curses window and advance cursor, 199  
 mvwaddnwstr — add a wide-character string to a window, 51  
 mvwaddstr — add a multi-byte character string (without rendition) to a window, 48  
 mvwaddwch — add a wchar\_t character (with attributes) to a curses window and advance cursor, 194  
 mvwaddwchnstr — add string of wchar\_t characters (and attributes) to a curses window, 197  
 mvwaddwchstr — add string of wchar\_t characters (and attributes) to a curses window, 197  
 mvwaddwstr — add a string of wchar\_t characters to a curses window and advance cursor, 199  
 mvwaddwstr — add a wide-character string to a window, 51  
 mvwchgat — change the rendition of characters in a window, 139  
 mvwdelch — remove a character, 319  
 mvwget\_wch — get a wide character from terminal, 486  
 mvwget\_wstr — get a wide character string from terminal, 472  
 mvwgetch — get a single-byte character from terminal, 462  
 mvwgetn\_wstr — get a wide character string from terminal, 472  
 mvwgetnstr — get a multibyte character string from terminal, 469  
 mvwgetnwstr — get wchar\_t character strings from curses terminal keyboard, 251  
 mvwgetstr — get a multibyte character string from terminal, 469  
 mvwgetwch — get (or push back) wchar\_t characters from curses terminal keyboard, 246  
 mvwgetwstr — get wchar\_t character strings from curses terminal keyboard, 251  
 mvwhline — use single-byte characters (and renditions) to draw lines, 513  
 mvwhline\_set — use complex characters (and renditions) to draw lines, 515  
 mvwin — move window, 944  
 mvwin\_wch — retrieve a complex character (with rendition), 593  
 mvwin\_wchnstr — retrieve complex character string (with rendition), 595  
 mvwin\_wchstr — retrieve complex character string (with rendition), 595  
 mvwinch — return a single-byte character (with rendition), 531  
 mvwinchnstr — retrieve a single-byte character string (with rendition), 533  
 mvwinchstr — retrieve a single-byte character string (with rendition), 533  
 mvwinnstr — retrieve a multibyte character string (without rendition), 555  
 mvwinnwstr — get a string of wchar\_t characters from a curses window, 269  
 mvwinnwstr — retrieve a wide character string (without rendition), 558  
 mvwins\_nwstr — insert a wide character string, 572  
 mvwins\_wch — insert a complex character, 582  
 mvwins\_wstr — insert a wide character string, 572  
 mvwinsch — insert a character, 561  
 mvwinsnstr — insert a multibyte character string, 568  
 mvwinsnwstr — insert wchar\_t string before character under the cursor in a curses window, 265  
 mvwinsstr — insert a multibyte character string, 568  
 mvwinstr — retrieve a multibyte character string (without rendition), 555  
 mvwinswch — insert a wchar\_t character before the character under the cursor in a curses window, 264  
 mvwinswstr — insert wchar\_t string before character under the cursor in a curses window, 265  
 mvwinwch — get a wchar\_t character and its attributes from a curses window, 267  
 mvwinwchnstr — get a string of wchar\_t characters (and attributes) from a curses window, 268  
 mvwinwchstr — get a string of wchar\_t characters (and attributes) from a curses window, 268

mvwinwstr — get a string of wchar\_t characters from a curses window, 269  
mvwinwstr — retrieve a wide character string (without rendition), 558  
mvwprintw — print formatted output window, 862  
mvwscanw — convert formatted input from a window, 864  
mvwvline — use single-byte characters (and renditions) to draw lines, 513  
mvwvline\_set — use complex characters (and renditions) to draw lines, 515  
mwwaddnstr — add a multi-byte character string (without rendition) to a window, 48

## N

napms — sleep process for a specified length of time, 1004  
newpad — create and display curses pads, 277  
newpad — create or refresh a pad or subpad, 1013  
newterm — screen initialization functions, 552  
newwin — create a new window or subwindow, 339  
nl — enable/disable newline control, 1029  
nocbreak — set input mode controls, 138  
nodelay — set blocking or non-blocking read, 1037  
noecho — enable/disable terminal echo, 353  
nonl — enable/disable newline control, 1029  
noqiflush — control flush of input and output on interrupt, 1049  
noraw — set input mode controls, 138  
notimeout — set timed blocking or non-blocking read, 1057  
number of columns on terminal screen — COLS, 178  
number of lines on terminal screen — LINES, 668

## O

openpl — graphics interface, 1102  
openvt — graphics interface, 1102

output attributes to the terminal — vidattr, 1492  
output attributes to the terminal — vidputs, 1492  
output attributes to the terminal — vid\_attr, 1492  
output attributes to the terminal — vid\_puts, 1492  
overlay — copy overlapped windows, 1066  
overlay or overwrite any portion of window — copywin, 183  
overwrite — copy overlapped windows, 1066

## P

pair\_content — manipulate color information, 132  
PAIR\_NUMBER — manipulate color information, 132  
panel library  
  *See also* curses library  
panels — character based panels package, 1089  
panels, create and destroy  
  — del\_panel, 1088  
  — new\_panel, 1088  
  — panel\_new, 1088  
panels deck manipulation routines  
  — bottom\_panel, 1092  
  — hide\_panel, 1091  
  — panel\_hidden, 1091  
  — panel\_show, 1091  
  — panel\_top, 1092  
  — show\_panel, 1091  
  — top\_panel, 1092  
panels deck traversal primitives  
  — panel\_above, 1084  
  — panel\_below, 1084  
panels panel, associate application data  
  — panel\_userptr, 1094  
  — set\_panel\_userptr, 1094  
panels panel, get or set current window  
  — panel\_window, 1095  
  — replace\_panel, 1095  
panels virtual screen refresh routine  
  — panel\_update, 1093  
  — update\_panel, 1093

panels window on virtual screen, move  
 — `move_panel`, 1087  
 — `panel_move`, 1087

`pecho_wchar` — add character and refresh window, 1098

`pechochar` — add character and refresh window, 1098

`pechochar` — create and display curses pads, 277

`pechowchar` — create and display curses pads, 277

`plot` — graphics interface, 1102  
 Link Editor, 1103

`pnoutrefresh` — create and display curses pads, 277

`pnoutrefresh` — create or refresh a pad or subpad, 1013

`point` — graphics interface, 1102

`prefresh` — create and display curses pads, 277

`prefresh` — create or refresh a pad or subpad, 1013

`print formatted output window` —  
`mvprintw`, 862

`print formatted output window` —  
`mvwprintw`, 862

`print formatted output window` — `printw`, 862

`print formatted output window` —  
`wprintw`, 862

`print formatted output in window` —  
`vwprintw`, 1508

`print formatted output in window` —  
`vw_printw`, 1507

`print formatted output in window` —  
`vw_scanw`, 1510

`printw` — print formatted output window, 862

`push character back onto the input queue`  
 — `unget_wch`, 1469  
 — `ungetch`, 1469

`putp` — apply padding information and output string, 1126

`putwin` — read a window from, and write a window to, a file, 490

## Q

`qiflush` — control flush of input and output on interrupt, 1049

## R

`raw` — set input mode controls, 138

`read a window from, and write a window to, a file`  
 — `getwin`, 490  
 — `putwin`, 490

`redraw screen or portion of screen`  
 — `redrawwin`, 1141  
 — `wredrawln`, 1141

`redrawwin` — redraw screen or portion of screen, 1141

`refresh` — refresh windows and lines, 343

`refresh windows and lines`  
 — `doupdate`, 343  
 — `refresh`, 343  
 — `wnoutrefresh`, 343  
 — `wrefresh`, 343

`remove a character`  
 — `delch`, 319  
 — `mvdelch`, 319  
 — `mvwdelch`, 319  
 — `wdelch`, 319

`remove a line`  
 — `deleteln`, 327  
 — `wdeleteln`, 327

`reserve screen line for dedicated purpose` —  
`ripoffline`, 1163

`reset_prog_mode` — save/restore terminal modes, 311

`reset_shell_mode` — save/restore terminal modes, 311

`resetty` — restore/save terminal modes, 1154

`restartterm` — interfaces to the terminfo database, 324

`restore initial terminal environment`  
 — `endwin`, 364  
 — `isendwin`, 364

`restore/save terminal modes`  
 — `resetty`, 1154  
 — `savetty`, 1154

`retrieve a complex character (with rendition)`  
 — `in_wch`, 593  
 — `mvin_wch`, 593  
 — `mvwin_wch`, 593  
 — `win_wch`, 593

`retrieve a multibyte character string (without rendition)`  
 — `innstr`, 555

- retrieve a multibyte character string (without rendition) (Continued)
  - instr, 555
  - mvinnstr, 555
  - mvinstr, 555
  - mvwinstr, 555
  - winnstr, 555
  - winstr, 555
- retrieve a single-byte character string (with rendition)
  - inchnstr, 533
  - inchstr, 533
  - mvinchnstr, 533
  - mvinchstr, 533
  - mvwinchnstr, 533
  - mvwinchstr, 533
  - winchnstr, 533
  - winchstr, 533
- retrieve a wide character string (without rendition)
  - innwstr, 558
  - inwstr, 558
  - mvinnwstr, 558
  - mvinwstr, 558
  - mvwinnwstr, 558
  - mvwinwstr, 558
  - winnwstr, 558
  - winwstr, 558
- retrieve complex character string (with rendition)
  - in\_wchnstr, 595
  - in\_wchstr, 595
  - mvin\_wchnstr, 595
  - mvin\_wchstr, 595
  - mvwin\_wchnstr, 595
  - mvwin\_wchstr, 595
  - win\_wchnstr, 595
  - win\_wchstr, 595
- return a single-byte character (with rendition)
  - inch, 531
  - mvinch, 531
  - mvwinch, 531
  - winch, 531
- return character string used as key name
  - key\_name, 633
  - keyname, 633

- return current ERASE or KILL characters
  - erasechar, 372
  - erasewchar, 372
  - killchar, 372
  - killwchar, 372
- return full terminal type name —
  - longname, 674
- return terminal baud rate — baudrate, 98
- return the value of a terminfo capability
  - tigetflag, 1415
  - tigetnum, 1415
  - tigetstr, 1415
  - tparm, 1415
- return the value of the environmental variable TERM — termname, 1390
- ripoffline — reserve screen line for dedicated purpose, 1163

**S**

- save/restore terminal modes
  - def\_prog\_mode, 311
  - def\_shell\_mode, 311
  - reset\_prog\_mode, 311
  - reset\_shell\_mode, 311
- savetty — restore/save terminal modes, 1154
- scanw — convert formatted input from a window, 864
- scr\_dump — write screen contents to/from a file, 1173
- scr\_init — write screen contents to/from a file, 1173
- scr\_restore — write screen contents to/from a file, 1173
- scr\_set — write screen contents to/from a file, 1173
- screen initialization functions
  - initscr, 552
  - newterm, 552
- sclr — scroll a window, 1178
- scroll — scroll a window, 1178
- scroll a window
  - sclr, 1178
  - scroll, 1178
  - wsclr, 1178
- scrollok — set terminal output controls, 149

set or get the background character (and rendition) of window — `bkgd`, 102  
 set or get the background character (and rendition) of window — `bkgdset`, 102  
 set or get the background character (and rendition) of window — `getbkgd`, 102  
 set or get the background character (and rendition) of window — `wbkgd`, 102  
 set or get the background character (and rendition) of window — `wbkgdset`, 102  
 set a `cchar_t` type character from a wide character and rendition — `setcchar`, 1192  
 set blocking or non-blocking read — `nodelay`, 1037  
 set/clear window attributes  
   — `standend`, 1360  
   — `standout`, 1360  
   — `wstandend`, 1360  
   — `wstandout`, 1360  
 set `curterm` — interfaces to the terminfo database, 324  
 set input mode controls  
   — `cbreak`, 138  
   — `nocbreak`, 138  
   — `noraw`, 138  
   — `raw`, 138  
 set or get the background character (and rendition) of window using a complex character  
   — `bkgd`, 107  
   — `bkgdset`, 107  
   — `getbkgd`, 107  
   — `wbkgd`, 107  
   — `wbkgdset`, 107  
   — `wgetbkgd`, 107  
 set `term` — switch between terminals, 1275  
 set terminal output controls  
   — `clearok`, 149  
   — `idlok`, 149  
   — `leaveok`, 149  
   — `scrollok`, 149  
   — `setscreg`, 149  
   — `wsetscreg`, 149  
 set timed blocking or non-blocking read  
   — `notimeout`, 1057  
   — `timeout`, 1057  
   — `wtimeout`, 1057  
 set visibility of cursor — `curs_set`, 288  
 setcchar — set a `cchar_t` type character from a wide character and rendition, 1192  
 setscreg — set terminal output controls, 149  
 setupterm — interfaces to the terminfo database, 324  
 sleep process for a specified length of time — `napms`, 1004  
`slk_attr_off` — soft label functions, 1291  
`slk_attr_on` — soft label functions, 1291  
`slk_attr_set` — soft label functions, 1291  
`slk_attroff` — soft label functions, 1291  
`slk_attron` — soft label functions, 1291  
`slk_attrset` — soft label functions, 1291  
`slk_clear` — soft label functions, 1291  
`slk_color` — soft label functions, 1291  
`slk_init` — soft label functions, 1291  
`slk_label` — soft label functions, 1291  
`slk_noutrefresh` — soft label functions, 1291  
`slk_refresh` — soft label functions, 1291  
`slk_restore` — soft label functions, 1291  
`slk_set` — soft label functions, 1291  
`slk_touch` — soft label functions, 1291  
`slk_wset` — soft label functions, 1291  
 soft label functions — `slk_attroff`, 1291  
 soft label functions — `slk_attron`, 1291  
 soft label functions — `slk_attrset`, 1291  
 soft label functions — `slk_attr_off`, 1291  
 soft label functions — `slk_attr_on`, 1291  
 soft label functions — `slk_attr_set`, 1291  
 soft label functions — `slk_clear`, 1291  
 soft label functions — `slk_color`, 1291  
 soft label functions — `slk_init`, 1291  
 soft label functions — `slk_label`, 1291  
 soft label functions — `slk_noutrefresh`, 1291  
 soft label functions — `slk_refresh`, 1291  
 soft label functions — `slk_restore`, 1291  
 soft label functions — `slk_set`, 1291  
 soft label functions — `slk_touch`, 1291  
 soft label functions — `slk_wset`, 1291  
 space — graphics interface, 1102  
 specify source of screen size information — `use_env`, 1485  
 standend — curses character and window attribute control routines, 201  
 standend — set/clear window attributes, 1360  
 standout — curses character and window attribute control routines, 201  
 standout — set/clear window attributes, 1360

start\_color — manipulate color information, 132  
 stdscr — default window, 1370  
 subpad — create and display curses pads, 277  
 subpad — create or refresh a pad or subpad, 1013  
 subwin — create a new window or subwindow, 339  
 switch between terminals — set\_term, 1275  
 synchronize window with its parents or children  
   — syncok, 1383  
   — wcursyncup, 1383  
   — wsyncdown, 1383  
   — wsyncup, 1383  
 syncok — synchronize window with its parents or children, 1383

## T

term\_attrs — get supported terminal video attributes, 1387  
 termattrs — get supported terminal video attributes, 1387  
 termname — return the value of the environmental variable TERM, 1390  
 tgetent — emulate the termcap database, 1393  
 tgetflag — emulate the termcap database, 1393  
 tgetnum — emulate the termcap database, 1393  
 tgetstr — emulate the termcap database, 1393  
 tgoto — emulate the termcap database, 1393  
 tigetflag — return the value of a terminfo capability, 1415  
 tigetnum — return the value of a terminfo capability, 1415  
 tigetstr — return the value of a terminfo capability, 1415  
 timeout — set timed blocking or non-blocking read, 1057  
 touchline — control window refresh, 608  
 touchwin — control window refresh, 608  
 tparm — return the value of a terminfo capability, 1415  
 tputs — apply padding information and output string, 1126  
 typeahead — check for type-ahead characters, 1460

## U

unctrl — generate printable representation of a character, 1463  
 unget\_wch — push character back onto the input queue, 1469  
 ungetch — push character back onto the input queue, 1469  
 ungetwch — get (or push back) wchar\_t characters from curses terminal keyboard, 246  
 untouchwin — control window refresh, 608  
 use complex characters (and renditions) to draw borders  
   — border\_set, 115  
   — box\_set, 115  
   — wborder\_set, 115  
 use complex characters (and renditions) to draw lines  
   — hline\_set, 515  
   — mvhline\_set, 515  
   — mvvline\_set, 515  
   — mvwhline\_set, 515  
   — mvwvline\_set, 515  
   — vline\_set, 515  
   — whline\_set, 515  
   — wvline\_set, 515  
 use\_env — specify source of screen size information, 1485  
 use single-byte characters (and renditions) to draw lines  
   — hline, 513  
   — mvhline, 513  
   — mvvline, 513  
   — mvwhline, 513  
   — mvwvline, 513  
   — vline, 513  
   — whline, 513  
   — wvline, 513

## V

vid\_attr — output attributes to the terminal, 1492  
 vid\_puts — output attributes to the terminal, 1492  
 vidattr — output attributes to the terminal, 1492

vidputs — output attributes to the terminal, 1492  
vline — use single-byte characters (and renditions) to draw lines, 513  
vline\_set — use complex characters (and renditions) to draw lines, 515  
vwprintw — print formatted output in window, 1508  
vw\_printw — print formatted output in window, 1507  
vw\_scanw — print formatted output in window, 1510  
vwscanw — convert formatted input from a window, 1511

## W

wadd\_wch — add a complex character (with rendition) to a window, 59  
wadd\_wchnstr — copy a string of complex characters (with renditions) to a window, 63  
wadd\_wchstr — copy a string of complex characters (with renditions) to a window, 63  
waddch — add a character (with rendition) to a window, 39  
waddchnstr — copy a character string (with renditions) to a window, 45  
waddchstr — copy a character string (with renditions) to a window, 45  
waddnstr — add a multi-byte character string (without rendition) to a window, 48  
waddnwstr — add a string of wchar\_t characters to a curses window and advance cursor, 199  
waddnwstr — add a wide-character string to a window, 51  
waddstr — add a multi-byte character string (without rendition) to a window, 48  
waddwch — add a wchar\_t character (with attributes) to a curses window and advance cursor, 194  
waddwchnstr — add string of wchar\_t characters (and attributes) to a curses window, 197  
waddwchstr — add string of wchar\_t characters (and attributes) to a curses window, 197

waddwstr — add a string of wchar\_t characters to a curses window and advance cursor, 199  
waddwstr — add a wide-character string to a window, 51  
wadjcurspos — moving the cursor by character, 200  
wattr\_get — control window attributes, 76  
wattr\_off — control window attributes, 76  
wattr\_on — control window attributes, 76  
wattr\_set — control window attributes, 76  
wattroff — change foreground window attributes, 82  
wattroff — curses character and window attribute control routines, 201  
wattron — change foreground window attributes, 82  
wattron — curses character and window attribute control routines, 201  
wattrset — change foreground window attributes, 82  
wattrset — curses character and window attribute control routines, 201  
wbkgd — set or get the background character (and rendition) of window, 102  
wbkgdset — set or get the background character (and rendition) of window, 102  
wbkgdset — set or get the background character (and rendition) of window using a complex character, 107  
wbkgdset — set or get the background character (and rendition) of window using a complex character, 107  
wborder — add a single-byte border to a window, 113  
wborder\_set — use complex characters (and renditions) to draw borders, 115  
wchgat — change the rendition of characters in a window, 139  
wclear — clear a window, 145  
wclrtoebot — clear to the end of a window, 158  
wclrtoeol — clear to the end of a line, 160  
wcolor\_set — control window attributes, 76  
wcursyncup — synchronize window with its parents or children, 1383  
wdelch — remove a character, 319  
wdeleteln — remove a line, 327  
wecho\_wchar — add a complex character and refresh window, 361

wechochar — add a single-byte character and refresh window, 357  
 wechowchar — add a `wchar_t` character (with attributes) to a curses window and advance cursor, 194  
 werase — clear a window, 145  
 wget\_wch — get a wide character from terminal, 486  
 wget\_wstr — get a wide character string from terminal, 472  
 wgetbkgrnd — set or get the background character (and rendition) of window using a complex character, 107  
 wgetch — get a single-byte character from terminal, 462  
 wgetn\_wstr — get a wide character string from terminal, 472  
 wgetnstr — get a multibyte character string from terminal, 469  
 wgetnwstr — get `wchar_t` character strings from curses terminal keyboard, 251  
 wgetstr — get a multibyte character string from terminal, 469  
 wgetwch — get (or push back) `wchar_t` characters from curses terminal keyboard, 246  
 wgetwstr — get `wchar_t` character strings from curses terminal keyboard, 251  
 whline — use single-byte characters (and renditions) to draw lines, 513  
 whline\_set — use complex characters (and renditions) to draw lines, 515  
 win\_wch — retrieve a complex character (with rendition), 593  
 win\_wchnstr — retrieve complex character string (with rendition), 595  
 win\_wchstr — retrieve complex character string (with rendition), 595  
 winch — return a single-byte character (with rendition), 531  
 winchnstr — retrieve a single-byte character string (with rendition), 533  
 winchstr — retrieve a single-byte character string (with rendition), 533  
 winnstr — retrieve a multibyte character string (without rendition), 555  
 winnwstr — get a string of `wchar_t` characters from a curses window, 269  
 winnwstr — retrieve a wide character string (without rendition), 558  
 wins\_nwstr — insert a wide character string, 572  
 wins\_wch — insert a complex character, 582  
 wins\_wstr — insert a wide character string, 572  
 winsch — insert a character, 561  
 winsdelln — insert/delete lines to/from the window, 563  
 winsertln — insert a line in a window, 565  
 winsnstr — insert a multibyte character string, 568  
 winsnwstr — insert `wchar_t` string before character under the cursor in a curses window, 265  
 winsstr — insert a multibyte character string, 568  
 winstr — retrieve a multibyte character string (without rendition), 555  
 winswch — insert a `wchar_t` character before the character under the cursor in a curses window, 264  
 winswstr — insert `wchar_t` string before character under the cursor in a curses window, 265  
 winwch — get a `wchar_t` character and its attributes from a curses window, 267  
 winwchnstr — get a string of `wchar_t` characters (and attributes) from a curses window, 268  
 winwchstr — get a string of `wchar_t` characters (and attributes) from a curses window, 268  
 winwstr — get a string of `wchar_t` characters from a curses window, 269  
 winwstr — retrieve a wide character string (without rendition), 558  
 wmove — move cursor in window, 726  
 wmovenextch — moving the cursor by character, 200  
 wmoveprevch — moving the cursor by character, 200  
 wnoutrefresh — refresh windows and lines, 343  
 wprintw — print formatted output window, 862  
 wredrawln — redraw screen or portion of screen, 1141  
 wrefresh — refresh windows and lines, 343

write screen contents to/from a file  
— `scr_dump`, 1173  
— `scr_init`, 1173  
— `scr_restore`, 1173  
— `scr_set`, 1173  
`wscanw` — convert formatted input from a  
window, 864  
`wscrl` — scroll a window, 1178  
`wsetscrreg` — set terminal output controls, 149  
`wstandend` — curses character and window  
attribute control routines, 201  
`wstandend` — set/clear window  
attributes, 1360  
`wstandout` — curses character and window  
attribute control routines, 201  
`wstandout` — set/clear window  
attributes, 1360  
`wsyncdown` — synchronize window with its  
parents or children, 1383  
`wsyncup` — synchronize window with its  
parents or children, 1383  
`wtimeout` — set timed blocking or non-blocking  
read, 1057  
`wtouchln` — control window refresh, 608  
`wunctrl` — generate printable representation of  
a wide character, 1751  
`wvline` — use single-byte characters (and  
renditions) to draw lines, 513  
`wvline_set` — use complex characters (and  
renditions) to draw lines, 515