man pages section 1: User Commands
## Contents

**Preface**  17

**Introduction**  23
  Intro(1)  24

**User Commands**  27
  acctcom(1)  28
  adb(1)  31
  addbib(1)  32
  alias(1)  34
  allocate(1)  37
  amt(1)  39
  answerbook2(1)  40
  appcert(1)  41
  apptrace(1)  48
  apropos(1)  53
  ar(1)  55
  arch(1)  59
  as(1)  60
  asa(1)  64
  at(1)  66
  atq(1)  72
  atrm(1)  73
  audioconvert(1)  74
  audioplay(1)  78
audiorecord(1)  80
auths(1)    83
awk(1)      85
banner(1)   90
basename(1) 91
basename(1B)93
bc(1)       94
bdiff(1)    98
bfs(1)      99
biff(1B)    103
break(1)    104
cal(1)      106
calendar(1) 107
cancel(1)   109
cat(1)      111
cc(1B)      113
cd(1)       115
cdrw(1)     118
checknr(1)  124
chgrp(1)    125
chkey(1)    127
chmod(1)    129
chown(1)    135
chown(1B)   137
ckdate(1)   138
ckgid(1)    141
ckint(1)    143
ckitem(1)   145
ckkeywd(1)  148
ckpath(1)   150
ckrange(1)  153
ckstr(1)    156
cksum(1)    159
cktime(1)   161
ckuid(1)    163
ckyorn(1)   165
clear(1)    167
cmp(1)      168
col(1) 170
comm(1) 172
command(1) 174
compress(1) 177
coproc(1F) 180
cp(1) 184
cpio(1) 188
cpp(1) 196
cputrack(1) 202
crle(1) 206
crontab(1) 216
crypt(1) 220
csh(1) 221
csplit(1) 247
c(1C) 250
ctags(1) 252
cut(1C) 255
cut(1) 261
date(1) 264
dc(1) 268
deallocate(1) 272
deroff(1) 274
df(1B) 275
dhcpinfo(1) 276
diff(1) 278
diff3(1) 282
diffmk(1) 284
dircmp(1) 285
dis(1) 286
dispgid(1) 288
dispuid(1) 289
dos2unix(1) 290
download(1) 292
dpost(1) 294
du(1) 297
du(1B) 300
dump(1) 302
dumpcs(1) 305
echo(1) 306
echo(1B) 310
echo(1F) 311
ed(1) 312
edit(1) 324
egrep(1) 328
eject(1) 331
elfdump(1) 335
enable(1) 337
env(1) 339
eqn(1) 341
error(1) 346
ex(1) 350
exec(1) 359
exit(1) 361
expand(1) 363
exportfs(1B) 365
expr(1) 366
expr(1B) 369
exstr(1) 372
face(1) 376
factor(1) 377
fastboot(1B) 378
fdformat(1) 379
fgrep(1) 383
file(1) 385
file(1B) 387
filesync(1) 389
find(1) 396
finger(1) 403
fmlcut(1F) 406
fmlexpr(1F) 408
fmlgrep(1F) 410
fml(1) 412
fmt(1) 415
fmtmsg(1) 416
fnattr(1) 421
fnbind(1) 424
<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>fnlist</td>
<td>426</td>
</tr>
<tr>
<td>fnlookup</td>
<td>428</td>
</tr>
<tr>
<td>fnrename</td>
<td>429</td>
</tr>
<tr>
<td>fnsearch</td>
<td>430</td>
</tr>
<tr>
<td>fnunbind</td>
<td>436</td>
</tr>
<tr>
<td>fold</td>
<td>437</td>
</tr>
<tr>
<td>from(1B)</td>
<td>439</td>
</tr>
<tr>
<td>ftp</td>
<td>440</td>
</tr>
<tr>
<td>ftpcount</td>
<td>451</td>
</tr>
<tr>
<td>ftpwho</td>
<td>452</td>
</tr>
<tr>
<td>gcore</td>
<td>453</td>
</tr>
<tr>
<td>genca</td>
<td>454</td>
</tr>
<tr>
<td>geniconvtabl</td>
<td>457</td>
</tr>
<tr>
<td>genlayouttabl</td>
<td>460</td>
</tr>
<tr>
<td>genmsg</td>
<td>475</td>
</tr>
<tr>
<td>getconf</td>
<td>481</td>
</tr>
<tr>
<td>getfacl</td>
<td>486</td>
</tr>
<tr>
<td>getfrm(1F)</td>
<td>490</td>
</tr>
<tr>
<td>getitems(1F)</td>
<td>491</td>
</tr>
<tr>
<td>getopt</td>
<td>492</td>
</tr>
<tr>
<td>getoptcvt</td>
<td>494</td>
</tr>
<tr>
<td>getoptss</td>
<td>497</td>
</tr>
<tr>
<td>getext</td>
<td>503</td>
</tr>
<tr>
<td>gettxt</td>
<td>505</td>
</tr>
<tr>
<td>glob</td>
<td>507</td>
</tr>
<tr>
<td>gprof</td>
<td>508</td>
</tr>
<tr>
<td>graph</td>
<td>513</td>
</tr>
<tr>
<td>grep</td>
<td>515</td>
</tr>
<tr>
<td>groups</td>
<td>520</td>
</tr>
<tr>
<td>groups(1B)</td>
<td>521</td>
</tr>
<tr>
<td>grpck(1B)</td>
<td>522</td>
</tr>
<tr>
<td>hash</td>
<td>523</td>
</tr>
<tr>
<td>head</td>
<td>525</td>
</tr>
<tr>
<td>history</td>
<td>527</td>
</tr>
<tr>
<td>hostid</td>
<td>536</td>
</tr>
<tr>
<td>hostname</td>
<td>537</td>
</tr>
<tr>
<td>iconv</td>
<td>538</td>
</tr>
<tr>
<td>indicator</td>
<td>540</td>
</tr>
<tr>
<td>Command</td>
<td>Page</td>
</tr>
<tr>
<td>--------------------</td>
<td>------</td>
</tr>
<tr>
<td>nisrmdir(1)</td>
<td>1036</td>
</tr>
<tr>
<td>nistbladm(1)</td>
<td>1038</td>
</tr>
<tr>
<td>nistest(1)</td>
<td>1044</td>
</tr>
<tr>
<td>nl(1)</td>
<td>1046</td>
</tr>
<tr>
<td>nm(1)</td>
<td>1049</td>
</tr>
<tr>
<td>nohup(1)</td>
<td>1054</td>
</tr>
<tr>
<td>nroff(1)</td>
<td>1058</td>
</tr>
<tr>
<td>od(1)</td>
<td>1061</td>
</tr>
<tr>
<td>on(1)</td>
<td>1067</td>
</tr>
<tr>
<td>optisa(1)</td>
<td>1069</td>
</tr>
<tr>
<td>pack(1)</td>
<td>1070</td>
</tr>
<tr>
<td>pagesize(1)</td>
<td>1073</td>
</tr>
<tr>
<td>pargs(1)</td>
<td>1074</td>
</tr>
<tr>
<td>passwd(1)</td>
<td>1076</td>
</tr>
<tr>
<td>paste(1)</td>
<td>1082</td>
</tr>
<tr>
<td>patch(1)</td>
<td>1085</td>
</tr>
<tr>
<td>pathchk(1)</td>
<td>1090</td>
</tr>
<tr>
<td>pathconv(1F)</td>
<td>1093</td>
</tr>
<tr>
<td>pax(1)</td>
<td>1095</td>
</tr>
<tr>
<td>perl(1)</td>
<td>1104</td>
</tr>
<tr>
<td>pfexec(1)</td>
<td>1111</td>
</tr>
<tr>
<td>pg(1)</td>
<td>1112</td>
</tr>
<tr>
<td>pgrep(1)</td>
<td>1117</td>
</tr>
<tr>
<td>pkginfo(1)</td>
<td>1121</td>
</tr>
<tr>
<td>pkgmk(1)</td>
<td>1123</td>
</tr>
<tr>
<td>pkgparam(1)</td>
<td>1126</td>
</tr>
<tr>
<td>pkgproto(1)</td>
<td>1128</td>
</tr>
<tr>
<td>pkgtrans(1)</td>
<td>1130</td>
</tr>
<tr>
<td>plimit(1)</td>
<td>1132</td>
</tr>
<tr>
<td>plot(1B)</td>
<td>1134</td>
</tr>
<tr>
<td>pmap(1)</td>
<td>1136</td>
</tr>
<tr>
<td>postdaisy(1)</td>
<td>1143</td>
</tr>
<tr>
<td>postdmd(1)</td>
<td>1145</td>
</tr>
<tr>
<td>postio(1)</td>
<td>1147</td>
</tr>
<tr>
<td>postmd(1)</td>
<td>1150</td>
</tr>
<tr>
<td>postplot(1)</td>
<td>1153</td>
</tr>
<tr>
<td>postprint(1)</td>
<td>1155</td>
</tr>
<tr>
<td>postreverse(1)</td>
<td>1157</td>
</tr>
</tbody>
</table>
posttek(1)  1159
ppgsz(1)   1161
pr(1)      1164
praliases(1)  1168
prctl(1)   1169
preap(1)   1172
prex(1)   1174
print(1)   1185
printenv(1B)  1186
printf(1)  1187
priocntl(1)  1192
proc(1)   1203
prof(1)   1206
profiles(1)  1210
projects(1)  1212
ps(1)   1213
ps(1B)  1222
pvs(1)   1225
pwd(1)  1228
ranlib(1)  1229
rcp(1)   1230
rdist(1)  1232
read(1)   1237
readfile(1F)  1240
readonly(1)  1241
refer(1)   1242
regcmp(1)  1244
regex(1F)  1246
reinit(1F)  1248
renice(1)  1249
reset(1F)  1252
rlogin(1)  1253
rm(1)    1256
rmformat(1)  1260
roffbib(1)  1268
roles(1)  1270
rpcgen(1)  1272
rpm2cpio(1)  1277
rsh(1) 1278
run(1F) 1281
runat(1) 1283
rup(1) 1286
rup(1C) 1287
ruptime(1) 1288
rusage(1B) 1289
rusers(1) 1291
rwho(1) 1292
sag(1) 1293
sar(1) 1295
sccs(1) 1300
sccs-admin(1) 1310
sccs-cdc(1) 1314
sccs-comb(1) 1316
sccs-delta(1) 1318
sccs-get(1) 1321
sccs-help(1) 1327
sccs-prs(1) 1328
sccs-prt(1) 1332
sccs-rmdel(1) 1335
sccs-sact(1) 1336
sccs-sccsdiff(1) 1337
sccs-unget(1) 1338
sccs-val(1) 1339
scp(1) 1341
script(1) 1343
sdiff(1) 1344
sed(1) 1346
sed(1B) 1353
set(1) 1359
set(1F) 1364
setcolor(1F) 1366
setfacl(1) 1367
setpgp(1) 1371
sftp(1) 1372
sh(1) 1375
shell(1F) 1393
<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>telnet(1)</td>
<td>1508</td>
</tr>
<tr>
<td>test(1)</td>
<td>1518</td>
</tr>
<tr>
<td>test(1B)</td>
<td>1526</td>
</tr>
<tr>
<td>test(1F)</td>
<td>1528</td>
</tr>
<tr>
<td>tftp(1)</td>
<td>1530</td>
</tr>
<tr>
<td>time(1)</td>
<td>1533</td>
</tr>
<tr>
<td>times(1)</td>
<td>1536</td>
</tr>
<tr>
<td>timex(1)</td>
<td>1537</td>
</tr>
<tr>
<td>tip(1)</td>
<td>1539</td>
</tr>
<tr>
<td>tnfdump(1)</td>
<td>1548</td>
</tr>
<tr>
<td>tnfxtract(1)</td>
<td>1553</td>
</tr>
<tr>
<td>touch(1)</td>
<td>1555</td>
</tr>
<tr>
<td>touch(1B)</td>
<td>1558</td>
</tr>
<tr>
<td>tplot(1)</td>
<td>1559</td>
</tr>
<tr>
<td>tput(1)</td>
<td>1560</td>
</tr>
<tr>
<td>tr(1)</td>
<td>1564</td>
</tr>
<tr>
<td>tr(1B)</td>
<td>1569</td>
</tr>
<tr>
<td>trap(1)</td>
<td>1570</td>
</tr>
<tr>
<td>troff(1)</td>
<td>1572</td>
</tr>
<tr>
<td>true(1)</td>
<td>1575</td>
</tr>
<tr>
<td>truss(1)</td>
<td>1576</td>
</tr>
<tr>
<td>tset(1B)</td>
<td>1583</td>
</tr>
<tr>
<td>tsort(1)</td>
<td>1588</td>
</tr>
<tr>
<td>tty(1)</td>
<td>1590</td>
</tr>
<tr>
<td>type(1)</td>
<td>1591</td>
</tr>
<tr>
<td>typeset(1)</td>
<td>1592</td>
</tr>
<tr>
<td>ucblinks(1B)</td>
<td>1594</td>
</tr>
<tr>
<td>ul(1)</td>
<td>1595</td>
</tr>
<tr>
<td>umask(1)</td>
<td>1596</td>
</tr>
<tr>
<td>uname(1)</td>
<td>1599</td>
</tr>
<tr>
<td>unifdef(1)</td>
<td>1602</td>
</tr>
<tr>
<td>uniq(1)</td>
<td>1604</td>
</tr>
<tr>
<td>units(1)</td>
<td>1606</td>
</tr>
<tr>
<td>unix2dos(1)</td>
<td>1608</td>
</tr>
<tr>
<td>uptime(1)</td>
<td>1610</td>
</tr>
<tr>
<td>users(1B)</td>
<td>1611</td>
</tr>
<tr>
<td>uucp(1C)</td>
<td>1612</td>
</tr>
<tr>
<td>uuencode(1C)</td>
<td>1616</td>
</tr>
</tbody>
</table>
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 ioctl(2) system call is called ioctl and generates its own heading. ioctl calls for a specific device are listed alphabetically (on the man page for that specific device). ioctl calls are used for a particular class of devices all of which have an io ending, such as mtcio(7I).

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 errno 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 `example%`, or if the user must be superuser, `example#`. 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 `attributes(5)` for more information.

**SEE ALSO**

This section lists references to other man pages, in-house documentation, and outside publications.
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIAGNOSTICS</td>
<td>This section lists diagnostic messages with a brief explanation of the condition causing the error.</td>
</tr>
<tr>
<td>WARNINGS</td>
<td>This section lists warnings about special conditions which could seriously affect your working conditions. This is not a list of diagnostics.</td>
</tr>
<tr>
<td>NOTES</td>
<td>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.</td>
</tr>
<tr>
<td>BUGS</td>
<td>This section describes known bugs and, wherever possible, suggests workarounds.</td>
</tr>
</tbody>
</table>
Introduction
This section describes, in alphabetical order, commands available with this operating system.

Pages of special interest are categorized as follows:

1B Commands found only in the SunOS/BSD Compatibility Package.
1C Commands for communicating with other systems.
1F Commands associated with Form and Menu Language Interpreter (FMLI).
1S Commands specific to the SunOS system.

See these sections of the man pages section 1M: System Administration Commands for more information:

- Section 1M in this manual for system maintenance commands.
- Section 4 of this manual for information on file formats.
- Section 5 of this manual for descriptions of publicly available files and miscellaneous information pages.
- Section 6 in this manual for computer demonstrations.

For tutorial information about these commands and procedures, see:

- Solaris Advanced User’s Guide

Unless otherwise noted, commands described in the SYNOPSIS section of a manual page accept options and other arguments according to the following syntax and should be interpreted as explained below.

```plaintext
name [-option...] [cmdarg...] where:

[ ] Surround an option or cmdarg that is not required.
...
Indicates multiple occurrences of the option or cmdarg.

name
The name of an executable file.

{ } The options and/or arguments enclosed within braces are interdependent, such that everything enclosed must be treated as a unit.

option (Always preceded by a “−”) noargletter... or, argletter optarg[,...]

noargletter A single letter representing an option without an option-argument. Note that more than one noargletter option can be grouped after one “−” (Rule 5, below).

argletter A single letter representing an option requiring an option-argument.
```
An option-argument (character string) satisfying a preceding argletter. Note that groups of optarg following an argletter must be separated by commas, or separated by a tab or space character and quoted (Rule 8, below).

cmdarg

Path name (or other command argument) not beginning with “−”, or “−” by itself indicating the standard input.

These command syntax rules are not followed by all current commands, but all new commands will obey them. getopta(1) should be used by all shell procedures to parse positional parameters and to check for legal options. It supports Rules 3-10 below. The enforcement of the other rules must be done by the command itself.

1. Command names (name above) must be between two and nine characters long.
2. Command names must include only lower-case letters and digits.
3. Option names (option above) must be one character long.
4. All options must be preceded by “−”.
5. Options with no arguments may be grouped after a single “−”.
6. The first option-argument (optarg above) following an option must be preceded by a tab or space character.
7. Option-arguments cannot be optional.
8. Groups of option-arguments following an option must either be separated by commas or separated by tab or space character and quoted (“−o xxx, z, yy or “−o "xxx z yy")).
9. All options must precede operands (cmdarg above) on the command line.
10. “−−” may be used to indicate the end of the options.
11. The order of the options relative to one another should not matter.
12. The relative order of the operands (cmdarg above) may affect their significance in ways determined by the command with which they appear.
13. “−” preceded and followed by a space character should only be used to mean standard input.

ATTRIBUTES

See attributes(5) for a discussion of the attributes listed in this section.

SEE ALSO
gopts(1), wait(1), exit(2), getopt(3C), wait(3UCB), attributes(5)

DIAGNOSTICS

Upon termination, each command returns two bytes of status, one supplied by the system and giving the cause for termination, and (in the case of “normal” termination) one supplied by the program [see wait(3UCB) and exit(2)]. The former byte is 0 for normal termination; the latter is customarily 0 for successful execution and non-zero to indicate troubles such as erroneous parameters, or bad or inaccessible data. It is called variously “exit code”, “exit status”, or “return code”, and is described only where special conventions are involved.
Some commands produce unexpected results when processing files containing null characters. These commands often treat text input lines as strings and therefore become confused upon encountering a null character (the string terminator) within a line.
User Commands
acctcom(1)

NAME
    acctcom – search and print process accounting files

SYNOPSIS
    acctcom [-abfhikmqrtv] [-C sec] [-e time] [-E time] [-g group]
            [-s time] [-S time] [-u user] [filename...]

DESCRIPTION
    The acctcom utility reads filenames, the standard input, or /var/adm/pacct, in the
    form described by acct(3HEAD) and writes selected records to standard output. Each
    record represents the execution of one process. The output shows the COMMAND NAME,
    USER, TTYPNAME, START TIME, END TIME, REAL (SEC), CPU (SEC), MEAN SIZE
    (K), and optionally, F (the fork() / exec() flag: 1 for fork() without exec()),
    STAT (the system exit status), HOG FACTOR, KCORE MIN, CPU FACTOR, CHARS
    TRNSFD, and BLOCKS READ (total blocks read and written).

    A ‘#’ is prepended to the command name if the command was executed with
    super-user privileges. If a process is not associated with a known terminal, a ‘?’ is
    printed in the TTYPNAME field.

    If no filename is specified, and if the standard input is associated with a terminal or
    /dev/null (as is the case when using ‘&’ in the shell), /var/adm/pacct is read;
    otherwise, the standard input is read.

    If any filename arguments are given, they are read in their respective order. Each file is
    normally read forward, that is, in chronological order by process completion time. The
    file /var/adm/pacct is usually the current file to be examined; a busy system may
    need several such files of which all but the current file are found in
    /var/adm/pacctincr.

OPTIONS
    The following options are supported:

    -a  Show some average statistics about the processes selected. The
        statistics will be printed after the output records.

    -b  Read backwards, showing latest commands first. This option has
        no effect when standard input is read.

    -f  Print the fork() / exec() flag and system exit status columns in
        the output. The numeric output for this option will be in octal.

    -h  Instead of mean memory size, show the fraction of total available
        CPU time consumed by the process during its execution. This “hog
        factor” is computed as (total CPU time)/(elapsed time).

    -i  Print columns containing the I/O counts in the output.

    -k  Instead of memory size, show total kcore-minutes.

    -m  Show mean core size (the default).

    -q  Do not print any output records, just print the average statistics as
        with the -a option.

    -r  Show CPU factor (user-time/(system-time + user-time)).
acctcom(1)

- \( t \)  
  Show separate system and user CPU times.

- \( v \)  
  Exclude column headings from the output.

- \( c \)  \( \text{sec} \)  
  Show only processes with total CPU time (system-time + user-time) exceeding \( \text{sec} \) seconds.

- \( e \)  \( \text{time} \)  
  Select processes existing at or before \( \text{time} \).

- \( E \)  \( \text{time} \)  
  Select processes ending at or before \( \text{time} \). Using the same \( \text{time} \) for both -\( S \) and -\( E \) shows the processes that existed at \( \text{time} \).

- \( g \)  \( \text{group} \)  
  Show only processes belonging to \( \text{group} \). The \( \text{group} \) may be designated by either the group ID or group name.

- \( H \)  \( \text{factor} \)  
  Show only processes that exceed \( \text{factor} \), where \( \text{factor} \) is the "hog factor" as explained in option -\( h \) above.

- \( I \)  \( \text{chars} \)  
  Show only processes transferring more characters than the cutoff number given by \( \text{chars} \).

- \( l \)  \( \text{line} \)  
  Show only processes belonging to terminal /dev/term/line.

- \( n \)  \( \text{pattern} \)  
  Show only commands matching \( \text{pattern} \) that may be a regular expression as in regcmp(3C), except + means one or more occurrences.

- \( o \)  \( \text{output-file} \)  
  Copy selected process records in the input data format to \( \text{output-file} \); suppress printing to standard output.

- \( o \)  \( \text{sec} \)  
  Show only processes with CPU system time exceeding \( \text{sec} \) seconds.

- \( s \)  \( \text{time} \)  
  Select processes existing at or after \( \text{time} \), given in the format \( \text{hr} [ :\text{min} [ :\text{sec} ] ] \).

- \( S \)  \( \text{time} \)  
  Select processes starting at or after \( \text{time} \).

- \( u \)  \( \text{user} \)  
  Show only processes belonging to \( \text{user} \). The user may be specified by a user ID, a login name that is then converted to a user ID, ‘#’ (which designates only those processes executed with superuser privileges), or ‘?’ (which designates only those processes associated with unknown user IDs).

FILES

/etc/group  
  system group file

/etc/passwd  
  system password file

/var/adm/pacctincr  
  active processes accounting file

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWaccu</td>
</tr>
</tbody>
</table>
acctcom(1)

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

SEE ALSO
ps(1), acct(1M), acctcms(1M), acctcon(1M), acctmerg(1M), acctprc(1M), acctsh(1M), fwtmp(1M), runacct(1M), su(1M), acct(2), regcmp(3C), acct(3HEAD), utmp(4), attributes(5)

System Administration Guide: Basic Administration

NOTES
acctcom reports only on processes that have terminated; use ps(1) for active processes.
adb – general-purpose debugger

SYNOPSIS

adb [-kw] [-I dir] [-P prompt] [-V mode] [object [core]]

DESCRIPTION

The adb utility is an interactive, general-purpose debugger. It can be used to examine files and provides a controlled environment for the execution of programs.

The adb utility is now implemented as a link to the mdb(1) utility in Solaris 9. mdb(1) is a low-level debugging utility that can be used to examine user processes as well as the live operating system or operating system crash dumps. The new mdb(1) utility provides complete backwards compatibility with the existing syntax and features of adb, including support for processing adb macro files. The Solaris Modular Debugger Guide and mdb(1) man page describes the features of mdb, including its adb compatibility mode. This mode will be activated by default when the adb link is executed.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWmdb (32-bit)</td>
</tr>
<tr>
<td></td>
<td>SUNWmdbx (64-bit)</td>
</tr>
</tbody>
</table>

SEE ALSO

mdb(1), attributes(5)

Solaris Modular Debugger Guide
addbib(1)

NAME  addbib – create or extend a bibliographic database

SYNOPSIS  addbib [-a] [-p promptfile] database

DESCRIPTION  When addbib starts up, answering y to the initial Instructions? prompt yields directions. Typing n (or RETURN) skips the directions. addbib then prompts for various bibliographic fields, reads responses from the terminal, and sends output records to database. A null response (just RETURN) means to leave out that field. A ‘−’ (minus sign) means to go back to the previous field. A trailing backslash allows a field to be continued on the next line. The repeating Continue? prompt allows the user either to resume by typing y (or RETURN), to quit the current session by typing n or q, or to edit database with any system editor (see vi(1), ex(1), ed(1)).

OPTIONS  The following options are supported:

- a  Suppresses prompting for an abstract. Asking for an abstract is the default. Abstracts are ended with a Control-D.

- p promptfile  Uses a new prompting skeleton, defined in promptfile. This file should contain prompt strings, a TAB, and the key-letters to be written to the database.

Bibliography Key Letters  The most common key-letters and their meanings are given below. addbib insulates you from these key-letters, since it gives you prompts in English, but if you edit the bibliography file later on, you will need to know this information.

%A  Author's name
%B  Book containing article referenced
%C  City (place of publication)
%D  Date of publication
%E  Editor of book containing article referenced
%F  Footnote number or label (supplied by refer)
%G  Government order number
%H  Header commentary, printed before reference
%I  Issuer (publisher)
%J  Journal containing article
%K  Keywords to use in locating reference
%L  Label field used by -k option of refer
%M  Bell Labs Memorandum (undefined)
%N  Number within volume
%O  Other commentary, printed at end of reference
%P  Page number(s)
EXAMPLE 1 Editing the bibliography file

Except for A, each field should be given just once. Only relevant fields should be supplied.

%A Mark Twain
%T Life on the Mississippi
%I Penguin Books
%C New York
%D 1978

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWdoc</td>
</tr>
</tbody>
</table>

SEE ALSO

ed(1), ex(1), indxbib(1), lookbib(1), refer(1), roffbib(1), sortbib(1), vi(1), attributes(5)
 alias, unalias – create or remove a pseudonym or shorthand for a command or series of commands

/usr/bin/alias [alias-name [= string...]]
/usr/bin/unalias alias-name...
/usr/bin/unalias -a

csh alias [name [def]]
unalias pattern

ksh alias [-tx] [name [= value]...]
unalias name...

DESCRIPTION
The alias and unalias utilities create or remove a pseudonym or shorthand term for a command or series of commands, with different functionality in the C-shell and Korn shell environments.

/usr/bin/alias
The alias utility creates or redefines alias definitions or writes the values of existing alias definitions to standard output. An alias definition provides a string value that replaces a command name when it is encountered.

An alias definition affects the current shell execution environment and the execution environments of the subshells of the current shell. When used as specified by this document, the alias definition will not affect the parent process of the current shell nor any utility environment invoked by the shell.

/usr/bin/unalias
The unalias utility removes the definition for each alias name specified. The aliases are removed from the current shell execution environment.

csh alias assigns def to the alias name. def is a list of words that may contain escaped history-substitution metasyntax. name is not allowed to be alias or unalias. If def is omitted, the alias name is displayed along with its current definition. If both name and def are omitted, all aliases are displayed.

Because of implementation restrictions, an alias definition must have been entered on a previous command line before it can be used.

unalias discards aliases that match (filename substitution) pattern. All aliases may be removed by ‘unalias *’.

ksh alias with no arguments prints the list of aliases in the form name=value on standard output. An alias is defined for each name whose value is given. A trailing space in value causes the next word to be checked for alias substitution. The -t flag is used to set and list tracked aliases. The value of a tracked alias is the full pathname corresponding to the given name. The value becomes undefined when the value of PATH is reset but the aliases remained tracked. Without the -t flag, for each name in
the argument list for which no value is given, the name and value of the alias is printed. The -x flag is used to set or print exported aliases. An exported alias is defined for scripts invoked by name. The exit status is non-zero if a name is given, but no value, and no alias has been defined for the name.

The aliases given by the list of names may be removed from the alias list with unalias.

**OPTIONS**

The following option is supported by unalias:

- `-a` Removes all alias definitions from the current shell execution environment.

**ksh**

The following option is supported by alias:

- `-t` Sets and lists tracked aliases.

**OPERANDS**

The following operands are supported:

**alias**

`alias-name` Write the alias definition to standard output.

**unalias**

`alias-name` The name of an alias to be removed.

`alias-name=string` Assign the value of string to the alias alias-name.

If no operands are given, all alias definitions will be written to standard output.

**OUTPUT**

The format for displaying aliases (when no operands or only name operands are specified) is:

```
"%s=%s\n" name, value
```

The value string will be written with appropriate quoting so that it is suitable for reinput to the shell.

**EXAMPLES**

**EXAMPLE 1** Modifying a command’s output

This example specifies that the output of the `ls` utility is columnated and more annotated:

```
example$ alias ls="ls -CF"
```

**EXAMPLE 2** Repeating previous entries in the command history file

This example creates a simple “redo” command to repeat previous entries in the command history file:

```
example$ alias r=’fc -s’
```

**EXAMPLE 3** Specifying a command’s output options

This example provides that the `du` utility summarize disk output in units of 1024 bytes:
EXAMPLE 3 Specifying a command’s output options  (Continued)

example% alias du=du -k

EXAMPLE 4 Dealing with an argument that is itself an alias name

This example sets up the nohup utility so that it can deal with an argument that is itself an alias name:

example% alias nohup="nohup *

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of alias and unalias: LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS

The following exit values are returned:

- 0 Successful completion.

- alias >0 One of the alias-name operands specified did not have an alias definition, or an error occurred.

- unalias >0 One of the alias-name operands specified did not represent a valid alias definition, or an error occurred.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO
csh(1), ksh(1), shell_builtins(1), attributes(5), environ(5)
allocate – device allocation

SYNOPSIS
allocate [-s] [-U uname] device
allocate [-s] [-U uname] -g dev-type
allocate [-s] [-U uname] -F device

DESCRIPTION
The allocate utility manages the ownership of devices through its allocation mechanism. It ensures that each device is used by only one qualified user at a time.

The device argument specifies the device to be manipulated. To preserve the integrity of the device’s owner, the allocate operation is executed on all the device special files associated with that device.

The argument dev-type is the device type to be operated on and can only be used with the -g option.

The default allocate operation allocates the device special files associated with device to the uid of the current process.

If the -F option is specified, the device cleaning program is executed when allocation is performed. This cleaning program is found in /etc/security/lib. The name of this program is found in the device_allocate(4) entry for the device in the dev-exec field.

Only authorized users may allocate a device. The required authorizations are specified in device_allocate(4).

OPTIONS
The following options are supported:

- -g dev-type Allocates a non–allocated device with a device–type matching dev-type.
- -s Silent. Suppresses any diagnostic output.
- -F device Reallocates the device allocated to another user. This option is often used with -U to reallocate a specific device to a specific user. Only a user with the solaris.devices.revoke authorization is permitted to use this option.
- -U uname Uses the user ID uname instead of the user ID of the current process when performing the allocate operation. Only a user with the solaris.devices.revoke authorization is permitted to use this option.

EXIT STATUS
The following exit values are returned:

- non–zero An error occurred.

FILES
/etc/security/device_allocate
/etc/security/device_maps
allocate(1)

/etc/security/dev/*
/etc/security/lib/*

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO
deallocate(1), list_devices(1), bsmconv(1M), dminfo(1M),
device_allocate(4), device_maps(4), attributes(5)

NOTES
The functionality described in this man page is available only if the Basic Security Module (BSM) has been enabled. See bsmconv(1M) for more information.
amt – run abstract machine test

SYNOPSIS  amt [-s]

DESCRIPTION  The amt command is for use in a Common Criteria security certified system. The command is used to verify that the low level functions necessary to enforce the object reuse requirements of the Controlled Access Protection Profile are working correctly. /usr/bin/amt is a shell script that executes tests specific to your system. For a 32-bit system, the tests run as a 32-bit application. For a 64-bit system, the tests run twice; once as a 32-bit application and once as a 64-bit application.

amt lists test results with a "pass" or "fail" for each test it performs, unless output is suppressed with the -s option.

OPTIONS  The following option is supported:

- s  Suppresses output.

EXIT STATUS  The following error values are returned:

- 0  All tests passed.
- >0  Count of the number of tests that failed.
- <0  Incorrect command line argument.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu (32-bit), SUNWcsxu (64-bit)</td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Evolving</td>
</tr>
</tbody>
</table>

SEE ALSO  attributes(5)
answerbook2(1)

NAME    answerbook2 – online documentation system

SYNOPSIS /usr/dt/bin/answerbook2 [-h]

DESCRIPTION The AnswerBook2 server product is no longer included with Solaris or the Solaris Documentation CD products. Solaris documentaion is now provided in HTML and PDF format on the Documentation CD and does not require the AnswerBook2 server to be viewed.

The answerbook2 utility opens the default web browser and displays an HTML page that shows a link to locally installed documentation, and, if the AnswerBook2 server has been defined, a link to AnswerBook2 collections.

To define a default AnswerBook2 server, use the environment variable, AB2_DEFAULTSERVER.

This functionality is also accessible through the AnswerBook2 option on the CDE front panel Help menu.

If you need an AnswerBook2 server, you can download the AnswerBook2 server software from http://www.sun.com.

OPTIONS The following option is supported:

- h       Displays a usage statement.

ENVIRONMENT VARIABLES AB2_DEFAULTSERVER  Fully-qualified URL that identifies the default AnswerBook2 server to use. For example:
                        http://imaserver.eng.sun.com:8888/

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td><a href="http://www.sun.com">http://www.sun.com</a></td>
</tr>
</tbody>
</table>

SEE ALSO attributes(5)

NOTES Use the online Help system to find out more about the AnswerBook2 product, once the web browser is opened and the AnswerBook2 library can be viewed.
appcert – examine application-level products for unstable use of Solaris interfaces

DESCRIPTION
The appcert utility examines an application’s conformance to the Solaris Application Binary Interface (ABI). The Solaris ABI defines the runtime library interfaces in Solaris that are safe and stable for application use. More specifically, appcert identifies any dependencies on unstable runtime interfaces, as well as certain other risks that could cause the product to fail to work on a subsequent release of Solaris.

appcert checks for:

- **Private symbol usage in Solaris libraries.** These are private symbols, that is, functions or data, that are not intended for developer consumption. They are interfaces that Solaris libraries use to call one another. These symbols might change their semantic behavior or even disappear altogether (so-called "demoted" symbols), so it is a good practice to make sure your application does not depend upon any of them.

- **Static linking.** In particular, this refers to static linking of archives libc.a, libsocket.a, and libnsl.a, that is, instead of dynamically linking the corresponding shared object .so’s. Because the semantics of private symbol calls from one Solaris library to another can change from one release to another, it is not a good practice to "hardwire" library code into your binary objects.

- **Unbound symbols.** These are library symbols (that is, functions or data) that the dynamic linker could not resolve when appcert was run. This might be an environment problem (for example, LD_LIBRARY_PATH) or a build problem (for example, not specifying -l lib and/or -z defs with compiling). They are flagged to point these problems out and in case a more serious problem is indicated.

An entire product can be readily examined by appcert (that is, if the product is a collection of many programs and supporting shared objects) by referring appcert to the directories where the product is installed.

To perform its task, appcert constructs a profile of interface dependencies for each object file within the product (whether an executable object or shared object), to determine all the Solaris system interfaces that are depended upon. (Notice that appcert uses the Solaris runtime linker to make this determination.) These dependency profiles are then compared to a definition of the Solaris ABI to identify any interfaces that are Private (unsafe and unstable for application-level use).

appcert generates a simple roll-up report that indicates which of the product’s components, if any, had liabilities and what those liabilities were. The report aids developers who are examining their product’s release-to-release stability.

Notice that appcert produces complete interface dependency information, both the Public (safe and stable) Solaris interfaces and the Private (non-ABI) interfaces. This information can also be examined for each product component, if you want.

IMPORTANT: appcert must run in the same environment in which the application being checked runs. See NOTES.
The following options are supported:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-B</code></td>
<td>If <code>appcert</code> is run in &quot;batch&quot; mode, the output report will contain one line per binary, beginning with <code>PASS</code> if no problems were detected for the binary, <code>FAIL</code> if any problems were found, or <code>INC</code> if the binary could not be completely checked. Do not interpret these labels too literally. For example, <code>PASS</code> just means that none of the <code>appcert</code> warnings were triggered. These strings are flush left and so can be selected via <code>grep</code> &quot;FAIL . . . , and so forth.</td>
</tr>
<tr>
<td><code>-f infile</code></td>
<td>Specifies the file <code>infile</code> that contains a list of files (one per line) to check. This list is appended to the list determined from the command line operands (see OPERANDS below).</td>
</tr>
<tr>
<td><code>-h</code></td>
<td>Prints out the usage information.</td>
</tr>
<tr>
<td><code>-L</code></td>
<td><code>appcert</code> examines your product for the presence of shared objects. If it finds some, it appends the directories they reside in to <code>LD_LIBRARY_PATH</code>. Use this flag to prevent <code>appcert</code> from doing this.</td>
</tr>
<tr>
<td><code>-n</code></td>
<td>When searching directories for binaries to check, this option does not follow symbolic links. See <code>find(1)</code>.</td>
</tr>
<tr>
<td><code>-S</code></td>
<td>Appends Solaris library directories (that is, <code>/usr/openwin/lib:/usr/dt/lib</code>) to <code>LD_LIBRARY_PATH</code>.</td>
</tr>
<tr>
<td><code>-w working_dir</code></td>
<td>Identifies the directory in which to run the library components and create temporary files (default is <code>/tmp</code>).</td>
</tr>
</tbody>
</table>

The following operands are supported:

<table>
<thead>
<tr>
<th>Operands</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>`{ obj</td>
<td>dir } ...`</td>
</tr>
</tbody>
</table>

The following exit values are returned:

<table>
<thead>
<tr>
<th>Exit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><code>appcert</code> ran successfully and found no potential binary stability problems.</td>
</tr>
<tr>
<td>1</td>
<td><code>appcert</code> failed to run successfully.</td>
</tr>
<tr>
<td>2</td>
<td>Some of the objects checked have potential binary stability problems.</td>
</tr>
<tr>
<td>3</td>
<td>No binary objects were located that could be checked.</td>
</tr>
</tbody>
</table>

If the object file to be examined depends on libraries, those dependencies must be recorded in it (by using the compiler's `-l` switch).

If the object file to be examined depends on other shared libraries, those libraries must be accessible via `LD_LIBRARY_PATH` or `RPATH` when `appcert` is run.
To check 64-bit applications, the machine must be running the 64-bit Solaris kernel. See `isalist(1)`. Also, the checks for static linking are currently not done on 64-bit applications.

`appcert` cannot examine:

- Object files that are completely or partially statically linked.

  Completely statically linked objects are reported as unstable.

- Executable files that do not have execute permission set.

  These are skipped. Shared objects without execute permission are not skipped.

- Object files that are setuid root.

  Due to limitations in `ldd(1)`, these are skipped. Copy and/or change the permissions to check them.

- Non-ELF file executables such as shell scripts.

- Non-C language interfaces to Solaris; for example, C++ and Java.

  The code itself need not be in C as long as the calls to Solaris libraries are in C.

**OUTPUT FILES**

`appcert` records its findings in the following files in the working directory (`/tmp/appcert???? by default):

- **Index**  
  A mapping between checked binaries and the subdirectory in the working directory in which the output specific to that binary can be found.

- **Report**  
  A copy of the rollup report that was displayed on stdout when `appcert` was run.

- **Skipped**  
  A list of binaries that `appcert` was asked to check but had to skip, along with a brief reason why each was skipped.

In addition, there is per-object information in the subdirectories under `appcert.????/objects/`, in the following files:

- **check.demoted_symbols**  
  A list of symbols suspected to be demoted Solaris symbols.

- **check.dynamic.private**  
  A list of private Solaris symbols to which the object makes direct bindings.

- **check.dynamic.public**  
  A list of public Solaris symbols to which the object makes direct bindings.
check.dynamic.unbound  A list of symbols not bound by the dynamic linker when `l礁d -r` was run. For convenience, `l礁d` output lines containing "file not found" are also included.

summary.dynamic  A pretty-printed summary of dynamic bindings for the objects examined, including tables of Public and Private symbols used from each Solaris library.

Other files are temporary files used internally by `appcert`.

**Private Symbol Use**

Private symbols are functions or data variables in a Solaris library that are not intended for developer or external use. These symbols are interfaces that the Solaris libraries use to call and communicate with one another. They are marked in `pvs(1)` output with the symbol version name "SUNWprivate".

Private symbols can change their semantic behavior or even disappear altogether ("demoted" or "deprecated" symbols), so your application should not depend upon any of them.

**Demoted Symbols**

Demoted symbols are functions or data variables in a Solaris library that were once private to that library and have been removed (or possibly scoped local to the library) in a later Solaris release. If your application directly calls one of these demoted symbols, it will fail to run (relocation error) on the release in which the symbol was removed and releases thereafter.

In some rare cases, a demoted symbol will return in a later release, but nevertheless there are still some releases on which the application will not run.

Sun Microsystems Inc. performed most of the library scoping in the transition from Solaris 2.5.1 to 2.6. This action was done to increase binary stability. By making these completely internal interfaces invisible (that is, they cannot be dynamically linked against), a developer cannot accidentally or intentionally call these interfaces. For more information, see the Linker and Libraries Guide, in particular the chapter on versioning. This document may be found online at [http://docs.sun.com](http://docs.sun.com).

**Unbound Symbols**

Unbound symbols are library symbols (that is, functions or data) referenced by the application that the dynamic linker could not resolve when `appcert` was run. *Note*: `appcert` does not actually run your application, so some aspect of the environment that affects dynamic linking might not be set properly.

Unbound symbols do not necessarily indicate a potential binary stability problem. They only mean that when `appcert` was run, the runtime dynamic dynamic linker could not resolve these symbols.

Unbound symbols might be due to `LD_LIBRARY_PATH` not being correctly set. Make sure it is set, so that all of your binary objects can find all of the libraries they depend on (either your product’s own libraries, Solaris libraries, or those of a third party). Then re-run `appcert`. 
You might find it useful to write a shell script that sets up the environment correctly and then runs appcert on the binaries you want to check.

Another common cause for unbound symbols is when a shared object under test has not recorded its dynamic dependencies, that is, at build time the -l switch was *not* supplied to the compiler and ld(1). So the shared object requires that the *executables* that link against it have the correct dependencies recorded.

Notice that such a shared object can either be linked in the standard way (that is, specified at an executable’s build time) or dynamically opened (for example, an executable calls dlopen(3DL) on the shared object sometimes when running). Either case can give rise to unbound symbols when appcert is run. The former can usually be resolved by setting LD_LIBRARY_PATH appropriately before running appcert. The latter (dlopen) is usually difficult to resolve. Under some circumstances, you might be able to set LD_PRELOAD appropriately to preload the needed libraries, but this procedure does not always work.

How do you know if the environment has been set up correctly so that there will be no unbound symbols? It must be set up so that running ldd -r on the binary yields no “file not found” or “symbol not found” errors. See ld.so.1(1) and ldd(1) for more information on dynamic linking.

In any event, appcert flags unbound symbols as a warning in case they might indicate a more serious problem. Unbound symbols can be an indicator of dependencies on demoted symbols (symbols that have been removed from a library or scoped local to it). Dependencies on demoted symbols will lead to serious binary stability problems.

However, setting up the environment properly should remove most unbound symbols. In general, it is good practice to record library dependencies at build time whenever possible because it helps make the binary object better defined and self-contained. Also recommended is using the -z defs flag when building shared objects, to force the resolution of all symbols during compilation. See ld(1) for more information.

appcert runs /bin/ldd -r on each binary object to be tested. It sets the environment variable LD_DEBUG=“files,bindings”. (See ldd(1) and ld.so.1(1) for more information). If that command fails for some reason, appcert will have no dynamic symbol binding information and will find “no bindings”.

appcert can fail if any of the following is true:

- The binary object does not have read permission.
- The binary object is SUID or SGID and the user does not have sufficient privileges.
- The binary object is an executable without the execute permission bit set.
- The binary object is a 64-bit application, but the kernel running on the current machine supports only 32-bit applications.
- The binary object is completely statically linked.
The binary object has no library dependency information recorded.

Other cases exist as well (for example, out of memory). In general, this flag means that appcert could not completely examine the object due to permissions or environment. Try to modify the permissions or environment so that the dynamic bindings can be recorded.

## Obsolete Library

An obsolete library is one whose use is deprecated and that might, in some future release, be removed from Solaris altogether. appcert flags these because applications depending on them might not run in future releases of Solaris. All interfaces, including Private ones, in an obsolete library are frozen and will not change.

## Use of sys_errlist/sys_nerr

Direct use of the symbols `sys_errlist` or `sys_nerr` presents a risk in which reference might be made past the end of the `sys_errlist` array. These symbols are deprecated in 32-bit versions of Solaris and are absent altogether in 64-bit versions. Use `strerror(3C)` instead.

## Use of Strong vs. Weak Symbols

The “strong” symbols (for example, `_socket`) associated with “weak” symbols (for example, `socket`) are reserved as private (their behavior could change in the future). Your application should only directly reference the weak symbol (usually the strong symbols begin with ` `_` `).

*Note:* Under certain build environments, the strong/private symbol dependency gets recorded into your binary instead of the weak/public one, even though the source code doesn’t appear to reference the private symbol. Nevertheless, steps should be taken to trace down why this is occurring and fix the dependency.

## NOTES

appcert needs to run in the same environment in which the application being checked runs. Otherwise it might not be able to resolve references correctly to interfaces in the Solaris libraries. Take the following steps:

1. Make sure that `LD_LIBRARY_PATH` and any other aspects of the environment are set to whatever settings are used when the application is run. Also make sure that it contains the directories containing any non-Solaris shared objects that are part of the product, so that they can be found when referenced.

2. Make sure that all the binaries to be checked:
   - Are dynamically linked ELF objects
   - Have execute permission set on executables (this is not necessary for shared objects)
   - Are not SUID root (otherwise you will have to be root to check them; make non-SUID copies and check those if necessary).

You might find it useful to write a shell script that sets up the environment correctly and then runs appcert.

Some potential problems that can be encountered are:

- appcert reports unbound symbols that appear to be part of Solaris libraries.
This is probably caused when the application uses `dlopen(3DL)` to access a shared object that does not have its Solaris dependencies recorded. `appcert` cannot resolve symbol use in such cases, since the dynamic linker is never invoked on the shared object, and there is no other dependency information that could be used to resolve the Solaris symbol bindings. This can also occur with non-Solaris symbols.

To avoid this problem, make sure that when a shared object is built, its dependencies on Solaris libraries are explicitly recorded by using the `-l` option on the compile line (see `cc(1)` and `ld(1)`).

- **appcert** reports that the application uses a Solaris private symbol that is not referenced in the application’s source code.

  This problem is most likely due to static linking of a Solaris library that references that symbol. Since `appcert` uses the dynamic linker to resolve symbols, statically linked libraries appear to `appcert` to be part of the application code (which, in a sense, they are). This can also sometimes happen as a result of macro substitution in a Solaris header file.

  To avoid this problem, whenever possible do not statically link Solaris library archives into your application.

- **appcert** does not recognize a library as part of Solaris.

  Some obsolete Solaris libraries are so old that they were obsoleted before their symbols could be versioned. Consequently, `appcert` cannot recognize them as being part of Solaris.

**BUGS**

The use of the terms “public” and “private” as equivalent to “stable” and “unstable” is unfortunately somewhat confusing. In particular, experimental or evolving interfaces are public in the sense that they are documented and their use is encouraged. But they are unstable, because an application built with them might not run on subsequent releases. Thus, they are classified as private for `appcert`’s purposes until they are no longer evolving. Conversely, obsolete interfaces will eventually disappear, and so are unstable, even though they have been public and stable in the past and are still treated as public by `appcert`. Fortunately, these two situations are rare.

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWapct</td>
</tr>
<tr>
<td>Interface stability</td>
<td>Stable</td>
</tr>
</tbody>
</table>

**SEE ALSO**

`cc(1), find(1), isalist(1), ld(1), ldd(1), ld.so.1(1), pvs(1), dlopen(3DL), strerror(3C), intro(4), attributes(5)`

*Linker and Libraries Guide*
apptrace(1)

NAME apptrace – trace application function calls to Solaris shared libraries

SYNOPSIS apptrace [-f] [-F [!] tracefromlist] [-T [!] tracetolist] [-o outputfile] [ [-tv] [!] call,...] command [command arguments]

DESCRIPTION The apptrace utility runs the executable program specified by command and traces all calls that the program command makes to the Solaris shared libraries. Tracing means that for each call the program makes, apptrace reports the name of the library interface called, the values of the arguments passed, and the return value.

By default, apptrace traces calls directly from the executable object to any of the shared objects it depends on. Indirect calls (that is, calls made between shared objects that the executable depends upon) are not reported by default.

Calls from or to additional shared objects may be traced using the -F or -T options (see below).

The default reporting format is a single line per call, with no formatted printing of arguments passed by reference or of data structures.

Formatted printing providing additional argument details is obtained using the -v option (see below).

By default, every interface provided by a shared object is traced if called. However, the set of interfaces to be traced can be restricted, using the -t and/or -v options.

Since it is generally possible to trace calls between any of the dynamic objects linked at runtime (the executable object and any of the shared objects depended upon), the report of each traced call gives the name of the object from which the call was made.

apptrace traces all of the procedure calls that occur between dynamic objects via the procedure linkage table, so only those procedure calls which are bound via the table will be traced. See the Linker and Libraries Guide.

OPTIONS The following options are supported:

- f Follows all children created by fork(2). This option will also cause the process id to be printed at the beginning of each line.

- F [!] tracefromlist Traces calls from a comma-separated list of shared objects. Only calls from these shared objects will be traced. The default is to trace calls from the main executable only. Only the basename of the shared object is required. For example, libc will match /usr/lib/libc.so.1. Additionally, shell style wildcard characters are supported as described in fnmatch(5). A list preceded by a “!” defines a list of objects from which calls should not be traced. If the tracing of calls from command is required, then command must be a member of tracefromlist.
apptrace(1)

- `outputfile` aptrace output will be directed to the `outputfile`. By
default, aptrace output is placed on the stderr
stream of the process being traced.

- `t [!]call,...` Traces or excludes function calls. Those calls specified
in the comma-separated list `call` are traced. If the list
begins with a `!`, the specified function calls are excluded
from the trace output. The default is `-t *`. The use of
shell style wildcards is allowed.

- `T [!]tracetolist` Traces calls to a comma-separated list of shared objects.
The default is to trace calls to all shared objects. As
above, the basename is all that is required and
wildcarding is allowed. A list preceded by a `''`
denotes a list of objects to which calls should not be
traced.

- `v [!]call,...` Provides verbose, formatted output of the arguments
and return values of the function calls specified (as
above in the `-t` option). Unlike `truss(1)`, calls named
by the `-v` option do not have to be named by the `-t`
option. For example, `apptrace -v open` is equivalent
to `truss -t open -v open`.

EXAMPLES

EXAMPLE 1 Tracing the date command

```
% apptrace date
date → libc.so.1:atexit(func = 0xff3ba1c8) = 0x0
date → libc.so.1:atexit(func = 0x117e4) = 0x0
date → libc.so.1:setlocale(category = 0x6, locale = "") = "C"
date → libc.so.1:textdomain(domainname = "SUNW_OST_OSCMD") = "SUNW_OST_OSCMD"
date → libc.so.1:getopt(argc = 0x1, argv = 0xffbeed5c,
   optstring = "a:u") = 0xffffffff errno = No error
date → libc.so.1:time(tloc = 0x21ecc) = 0x371397c3
date → libc.so.1:nl_langinfo(item = 0x3a) = "%a %b %e %T %Z %Y"
date → libc.so.1:localtime(clock = 0x21ecc) = 0xff03c928
date → libc_psr.so.1:memcpy(0xffbeeccc, 0xff03c928, 0x24)
date → libc.so.1:strftime(s = "Tue Apr 13 15:15:15 ",
   maxsize = 0x400, format = "%a %b %e %T %Z %Y",
   timeptr = 0xffbeeccc) = 0x1c
date → libc.so.1:puts(Tue Apr 13 15:15:15 EDT 1999
   s = "Tue Apr 13 15:15:15 ") = 0x1d
date → libc.so.1:exit(status = 0)
```

EXAMPLE 2 Tracing a specific set of interfaces with verbosity set

```
% apptrace -v 'gid*' id -a
id → libc.so.1:getgid() = 0xa
   return = (gid_t) 10 (0xa)
id → libc.so.1:getegid() = 0xa
   return = (gid_t) 10 (0xa)
id → libc.so.1:getgrgid(gid = 0xa) = 0x2238c
```
EXAMPLE 2 Tracing a specific set of interfaces with verbosity set (Continued)

```c
    gid = (gid_t) 10 (0xa)
    return = (struct group *) 0x2238c (struct group) {
        gr_name: (char *) 0x223a0 "staff"
        gr_passwd: (char *) 0x223a6 ""
        gr_gid: (gid_t) 10 (0xa)
        gr_mem: (char **) 0x2239c
    }

    id   -> libc.so.1:getgrgid(gid = 0xa) = 0x2238c
    gid = (gid_t) 10 (0xa)
    return = (struct group *) 0x2238c (struct group) {
        gr_name: (char *) 0x223a0 "staff"
        gr_passwd: (char *) 0x223a6 ""
        gr_gid: (gid_t) 10 (0xa)
        gr_mem: (char **) 0x2239c
    }

    id   -> libc.so.1:getgrgid(gid = 0x3) = 0x2238c
    gid = (gid_t) 3 (0x3)
    return = (struct group *) 0x2238c (struct group) {
        gr_name: (char *) 0x223b4 "sys"
        gr_passwd: (char *) 0x223b8 ""
        gr_gid: (gid_t) 3 (0x3)
        gr_mem: (char **) 0x2239c
    }

    id   -> libc.so.1:getgrgid(gid = 0x29) = 0x2238c
    gid = (gid_t) 41 (0x29)
    return = (struct group *) 0x2238c (struct group) {
        gr_name: (char *) 0x223a4 "opcom"
        gr_passwd: (char *) 0x223a8 ""
        gr_gid: (gid_t) 41 (0x29)
        gr_mem: (char **) 0x2239c
    }

    id   -> libc.so.1:getgrgid(gid = 0xe) = 0x2238c
    gid = (gid_t) 14 (0xe)
    return = (struct group *) 0x2238c (struct group) {
        gr_name: (char *) 0x223a0 "sysadmin"
        gr_passwd: (char *) 0x223a9 ""
        gr_gid: (gid_t) 14 (0xe)
        gr_mem: (char **) 0x2239c
    }

    id   -> libc.so.1:getgrgid(gid = 0xd3) = 0x2238c
    gid = (gid_t) 211 (0xd3)
    return = (struct group *) 0x2238c (struct group) {
        gr_name: (char *) 0x223a8 "test"
        gr_passwd: (char *) 0x223ad ""
        gr_gid: (gid_t) 211 (0xd3)
        gr_mem: (char **) 0x2239c
    }
```

uid=44013(georgn) gid=10(staff) groups=10(staff),3(sys),
EXAMPLE 2 Tracing a specific set of interfaces with verbosity set (Continued)

41(opcom),14(sysadmin),211(test)

FILES
Basic runtime support for apptrace is provided by the link auditing feature of the Solaris runtime linker (ld.so.1) and the apptrace command’s use of this facility relies on an auditing object (apptrace.so.1) kept in /usr/lib/abi.

In order to perform formatted printing of arguments when tracing calls (as selected by the -v option), apptrace needs to know the number and data types of the arguments supplied to the called interface. Special runtime support shared objects are provided which apptrace relies upon to perform formatted printing. A runtime support object is provided for each Solaris shared library, which contains an “interceptor” function for each interface within the shared library. These supporting shared objects are kept in /usr/lib/abi.apptrace has a simple algorithm to map from the name of a library interface to the name of an interceptor function in the library’s supporting verbose-tracing shared object. If an interceptor is not found in the library’s supporting tracing shared object, apptrace cannot determine either the number or data types of the arguments for that interface. In this case, apptrace uses a default output format for the call-tracing report (hex-formatted printing of the first three arguments).

LIMITATIONS
In general, apptrace cannot trace calls to functions accepting variable argument lists. There has been some clever coding in several specific cases to work around this limitation, most notably in the printf and scanf families.

Functions that attempt to probe the stack or otherwise extract information about the caller cannot be traced. Some examples are [gs]etcontext(), [sig]longjmp(), [sig]setjmp(), and vfork().

Functions such as exit(2) that do not return may also produce strange output. Also, functions that call other traced functions before returning will produce slightly garbled output.

For security reasons, only root can apptrace setuid/setgid programs.

Tracing functions whose usage requires the inclusion of varargs.h, such as vwprintw(3XCURSES) and vwsanw(3XCURSES), will not provide formatted printing of arguments.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcstl (32-bit)</td>
</tr>
<tr>
<td></td>
<td>SUNWcstlx (64-bit)</td>
</tr>
</tbody>
</table>
SEE ALSO
ld.so.1(1), truss(1), vwprintw(3XCURSES), vwscanw(3XCURSES),
attributes(5), fnmatch(5)

Linker and Libraries Guide
The `apropos` utility displays the man page name, section number, and a short description for each man page whose NAME line contains `keyword`. This information is contained in the `/usr/share/man/windex` database created by `catman(1M)`. If `catman(1M)` was not run, or was run with the `-n` option, `apropos` fails. Each word is considered separately and the case of letters is ignored. Words which are part of other words are considered; for example, when looking for 'compile', `apropos` finds all instances of 'compiler' also.

`apropos` is actually just the `-k` option to the `man(1)` command.

**EXAMPLE 1** To find a man page whose NAME line contains a keyword

Try

```
example% apropos password
```

and

```
example% apropos editor
```

If the line starts `filename(section) . . . ' you can run

```
man -s section filename
```

to display the man page for `filename`.

**EXAMPLE 2** To find the man page for the subroutine `printf()`

Try

```
example% apropos format
```

and then

```
example% man -s 3s printf
```

to get the manual page on the subroutine `printf()`.
DIAGNOSTICS

/usr/share/man/windex: No such file or directory
This database does not exist. catman(1M) must be run to create it.
NAME

ar - maintain portable archive or library

SYNOPSIS

/usr/ccs/bin/ar -d [-Vv] archive file...
/usr/ccs/bin/ar -m [-abiVv] [posname] archive file...
/usr/ccs/bin/ar -p [-svv] archive [file...]
/usr/ccs/bin/ar -q [-cvv] archive file...
/usr/ccs/bin/ar -r [-abcuvv] [posname] archive file...
/usr/ccs/bin/ar -t [-svv] archive [file...]
/usr/ccs/bin/ar -x [-CstTvVv] archive [file...]
/usr/xpg4/bin/ar -d [-Vv] archive file...
/usr/xpg4/bin/ar -m [-abiVv] [posname] archive file...
/usr/xpg4/bin/ar -p [-svv] archive [file...]
/usr/xpg4/bin/ar -q [-cvv] archive file...
/usr/xpg4/bin/ar -r [-abcuvv] [posname] archive file...
/usr/xpg4/bin/ar -t [-svv] archive [file...]
/usr/xpg4/bin/ar -x [-CstTvVv] archive [file...]

DESCRIPTION

The ar utility maintains groups of files combined into a single archive file. Its main use is to create and update library files. However, it can be used for any similar purpose. The magic string and the file headers used by ar consist of printable ASCII characters. If an archive is composed of printable files, the entire archive is printable.

When ar creates an archive, it creates headers in a format that is portable across all machines. The portable archive format and structure are described in detail in ar(3HEAD). The archive symbol table (described in ar(3HEAD)) is used by the link editor ld(1) to effect multiple passes over libraries of object files in an efficient manner. An archive symbol table is only created and maintained by ar when there is at least one object file in the archive. The archive symbol table is in a specially named file that is always the first file in the archive. This file is never mentioned or accessible to the user. Whenever the ar command is used to create or update the contents of such an archive, the symbol table is rebuilt. The -s option described below will force the symbol table to be rebuilt.

OPTIONS

The following options are supported:

-a Positions new files in archive after the file named by the posname operand.
-b Positions new files in archive before the file named by the posname operand.
-c Suppresses the diagnostic message that is written to standard error by default when archive is created.
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-C</code></td>
<td>Prevents extracted files from replacing like-named files in the file system. This option is useful when <code>-T</code> is also used to prevent truncated file names from replacing files with the same prefix.</td>
</tr>
<tr>
<td><code>-d</code></td>
<td>Deletes one or more files from <code>archive</code>.</td>
</tr>
<tr>
<td><code>-i</code></td>
<td>Positions new files in <code>archive</code> before the file named by the <code>posname</code> operand (equivalent to <code>-b</code>).</td>
</tr>
<tr>
<td><code>-m</code></td>
<td>Moves files. If <code>-a</code>, <code>-b</code>, or <code>-i</code> with the <code>posname</code> operand are specified, moves files to the new position; otherwise, moves files to the end of <code>archive</code>.</td>
</tr>
<tr>
<td><code>-p</code></td>
<td>Prints the contents of files in <code>archive</code> to standard output. If no files are specified, the contents of all files in <code>archive</code> will be written in the order of the archive.</td>
</tr>
<tr>
<td><code>-q</code></td>
<td>Quickly appends files to the end of <code>archive</code>. Positioning options <code>-a</code>, <code>-b</code>, and <code>-i</code> are invalid. The command does not check whether the added files are already in <code>archive</code>. This option is useful to avoid quadratic behavior when creating a large archive piece-by-piece.</td>
</tr>
<tr>
<td><code>-r</code></td>
<td>Replaces or adds files in <code>archive</code>. If <code>archive</code> does not exist, a new archive file will be created and a diagnostic message will be written to standard error (unless the <code>-c</code> option is specified). If no files are specified and the <code>archive</code> exists, the results are undefined. Files that replace existing files will not change the order of the archive. If the <code>-u</code> option is used with the <code>-r</code> option, then only those files with dates of modification later than the archive files are replaced. If the <code>-a</code>, <code>-b</code>, or <code>-i</code> option is used, then the <code>posname</code> argument must be present and specifies that new files are to be placed after (<code>-a</code>) or before (<code>-b</code> or <code>-i</code>) <code>posname</code>; otherwise the new files are placed at the end.</td>
</tr>
<tr>
<td><code>-s</code></td>
<td>Forces the regeneration of the archive symbol table even if <code>ar</code> is not invoked with a option which will modify the archive contents. This command is useful to restore the archive symbol table after the <code>strip(1)</code> command has been used on the archive.</td>
</tr>
<tr>
<td><code>-t</code></td>
<td>Prints a table of contents of <code>archive</code>. The files specified by the <code>file</code> operands will be included in the written list. If no file operands are specified, all files in <code>archive</code> will be included in the order of the archive.</td>
</tr>
<tr>
<td><code>-T</code></td>
<td>Allows file name truncation of extracted files whose archive names are longer than the file system can support. By default, extracting a file with a name that is too long is an error; a diagnostic message will be written and the file will not be extracted.</td>
</tr>
<tr>
<td><code>-u</code></td>
<td>Updates older files. When used with the <code>-r</code> option, files within <code>archive</code> will be replaced only if the corresponding <code>file</code> has a modification time that is at least as new as the modification time of the file within <code>archive</code>.</td>
</tr>
<tr>
<td><code>-V</code></td>
<td>Prints its version number on standard error.</td>
</tr>
<tr>
<td><code>/usr/bin/ar</code></td>
<td>Gives verbose output. When used with the option characters <code>-d</code>, <code>-r</code>, or <code>-x</code>, writes a detailed file-by-file description of the archive creation and the</td>
</tr>
</tbody>
</table>
constituent files, and maintenance activity. When used with -p, writes the
name of the file to the standard output before writing the file itself to the
standard output. When used with -t, includes a long listing of information
about the files within the archive. When used with -x, prints the filename
preceding each extraction. When writing to an archive, a message is written
to the standard error.

**OPERANDS**
The following operands are supported:

- `archive` A path name of the archive file.
- `file` A path name. Only the last component will be used when
  comparing against the names of files in the archive. If two or more
  file operands have the same last path name component (`basename(1)`), the results are unspecified. The implementation’s
  archive format will not truncate valid file names of files added to
  or replaced in the archive.
- `posname` The name of a file in the archive file, used for relative positioning;
  see options `-m` and `-r`.

**ENVIRONMENT VARIABLES**
See `environ(5)` for descriptions of the following environment variables that affect the
execution of `ar`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

**EXIT STATUS**
The following exit values are returned:

- `0` Successful completion.
- `>0` An error occurred.

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

```
/usr/bin/ar

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWbttool</td>
</tr>
</tbody>
</table>

/usr/xpg4/bin/ar

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWxcu4</td>
</tr>
</tbody>
</table>
```
SEE ALSO  basename(1), cc(1B), cpio(1), ld(1), lorder(1), strip(1), tar(1), ar(3HEAD),
a.out(4), attributes(5), environ(5), XPG4(5)

NOTES  If the same file is mentioned twice in an argument list, it may be put in the archive
twice.

By convention, archives are suffixed with the characters .a.
Name: arch – display the architecture of the current host

Synopsis: arch [-k | archname]

Description: arch displays the application architecture of the current host system. Due to extensive historical use of this command without any options, all SunOS 5.x SPARC based systems will return “sun4” as their application architecture. Use of this command is discouraged; see NOTES section below.

Systems can be broadly classified by their architectures, which define what executables will run on which machines. A distinction can be made between kernel architecture and application architecture (or, commonly, just “architecture”). Machines that run different kernels due to underlying hardware differences may be able to run the same application programs.

Options: -k Display the kernel architecture, such as sun4m, sun4c, and so forth. This defines which specific SunOS kernel will run on the machine, and has implications only for programs that depend on the kernel explicitly (for example, ps(1)).

Operands: The following operand is supported:

archname Use archname to determine whether the application binaries for this application architecture can run on the current host system. The archname must be a valid application architecture, such as sun4, i86pc, and so forth.

If application binaries for archname can run on the current host system, TRUE (0) is returned; otherwise, FALSE (1) is returned.

Exit Status: The following exit values are returned:

0 Successful completion.

>0 An error occurred.

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

<table>
<thead>
<tr>
<th>Attribute Type</th>
<th>Attribute Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

See Also: mach(1), ps(1), uname(1), attributes(5)

Notes: This command is provided for compatibility with previous releases and its use is discouraged. Instead, the uname command is recommended. See uname(1) for usage information.
as(1)

NAME

as – assembler


DESCRIPTION

The as command creates object files from assembly language source files.

Common Options

The following flags are common to both SPARC and IA. They may be specified in any order:

- **-b** Generates extra symbol table information for the Sun SourceBrowser.

- **-K PIC** Generates position-independent code.

- **-L** Saves all symbols, including temporary labels that are normally discarded to save space, in the ELF symbol table.

- **-m** Runs the m4(1) macro processor on the input to the assembler.

- **-n** Suppresses all the warnings while assembling.

- **-o outfile** Puts the output of the assembly in outfile. By default, the output file name is formed by removing the .s suffix, if there is one, from the input file name and appending a .o suffix.

- **-P** Runs cpp(1), the C preprocessor, on the files being assembled. The preprocessor is run separately on each input file, not on their concatenation. The preprocessor output is passed to the assembler.

- **-D name**

- **-D name=def** When the -P option is in effect, these options are passed to the cpp(1) preprocessor without interpretation by the as command; otherwise, they are ignored.

- **-I path** When the -P option is in effect, this option is passed to the cpp(1) preprocessor without interpretation by the as command; otherwise, it is ignored.
When the -U option is in effect, this option is passed to the cpp(1) preprocessor without interpretation by the as command; otherwise, it is ignored.

-Qu

If y is specified, this option produces the "assembler version" information in the comment section of the output object file. If n is specified, the information is suppressed.

-s

Places all stabs in the .stabs section. By default, stabs are placed in stabs.excl sections, which are stripped out by the static linker, ld(1), during final execution. When the -s option is used, stabs remain in the final executable because .stab sections are not stripped by the static linker.

-Sc

Produces a disassembly of the emitted code to the standard output. Adding each of the following characters to the -S option produces:

a disassembling with address
b disassembling with ".bof"
c disassembling with comments
l disassembling with line numbers

Capital letters turn the switch off for the corresponding option.

-T

This is a migration option for 4.x assembly files to be assembled on 5.x systems. With this option, the symbol names in 4.x assembly files will be interpreted as 5.x symbol names.

-V

Writes the version number of the assembler being run on the standard error output.

-xf

Generates additional information for performance analysis of the executable using Sun WorkShop analyzer. If the input file does not contain any stabs (debugging directives), then the assembler will generate some default stabs which are needed by the Sun WorkShop analyzer. Also see the dbx manual page available with Sun Workshop.

-q

Performs a quick assembly. When the -q option is used, many error checks are not performed. Note: This option disables many error checks. Use of this option to assemble handwritten assembly language is not recommended.
-xarch=v7
This option instructs the assembler to accept instructions defined in the SPARC version 7 (V7) architecture. The resulting object code is in ELF format.

-xarch=v8
This option instructs the assembler to accept instructions defined in the SPARC-V8 architecture, less the quad-precision floating-point instructions. The resulting object code is in ELF format.

-xarch=v8a
This option instructs the assembler to accept instructions defined in the SPARC-V8 architecture, less the quad-precision floating-point instructions and less the fsmul instruction. The resulting object code is in ELF format. This is the default choice of the -xarch=options.

-xarch=v8plus
This option instructs the assembler to accept instructions defined in the SPARC-V9 architecture, less the quad-precision floating-point instructions. The resulting object code is in ELF format. It will not execute on a Solaris V8 system (a machine with a V8 processor). It will execute on a Solaris V8+ system. This combination is a SPARC 64-bit processor and a 32-bit OS.

-xarch=v8plusa
This option instructs the assembler to accept instructions defined in the SPARC-V9 architecture, less the quad-precision floating-point instructions, plus the instructions in the Visual Instruction Set (VIS). The resulting object code is in V8+ ELF format. It will not execute on a Solaris V8 system (a machine with a V8 processor). It will execute on a Solaris V8+ system.

-xarch=v9
This option limits the instruction set to the SPARC-V9 architecture. The resulting .o object files are in 64-bit ELF format and can only be linked with other object files in the same format. The resulting executable can only be run on a 64-bit SPARC processor running 64-bit Solaris with the 64-bit kernel.

-xarch=v9a
This option limits the instruction set to the SPARC-V9 architecture, adding the Visual Instruction Set (VIS) and extensions specific to UltraSPARC processors. The resulting .o object files are in 64-bit ELF format and can only be linked with other object files in the same format. The resulting executable can only be run on a 64-bit SPARC processor running 64-bit Solaris with the 64-bit kernel.

OPERANDS
The following operand is supported:
The `as` command normally creates temporary files in the directory `/tmp`. Another directory may be specified by setting the environment variable `TMPDIR` to the chosen directory. (If `TMPDIR` is not a valid directory, then `as` will use `/tmp`).

By default, `as` creates its temporary files in `/tmp`.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsprot</td>
</tr>
</tbody>
</table>

If the `-m` option, which invokes the `m4(1)` macro processor, is used, keywords for `m4` cannot be used as symbols (variables, functions, labels) in the input file, since `m4` cannot determine which keywords are assembler symbols and which keywords are real `m4` macros.

Whenever possible, access the assembler through a compilation system interface program such as `cc(1B)`.

All undefined symbols are treated as global.
asa(1)

NAME
asa – convert FORTRAN carriage-control output to printable form

SYNOPSIS
asa [-f] [file...]

DESCRIPTION
The asa utility will write its input files to standard output, mapping carriage-control characters from the text files to line-printer control sequences.

The first character of every line will be removed from the input, and the following actions will be performed.

If the character removed is:

SPACE  The rest of the line will be output without change.

0      It is replaced by a newline control sequence followed by the rest of the input line.

1      It is replaced by a newpage control sequence followed by the rest of the input line.

+      It is replaced by a control sequence that causes printing to return to the first column of the previous line, where the rest of the input line is printed.

For any other character in the first column of an input line, asa skips the character and prints the rest of the line unchanged.

If asa is called without providing a filename, the standard input is used.

OPTIONS
The following option is supported:

-f      Start each file on a new page.

OPERANDS
The following operand is supported:

date   A pathname of a text file used for input. If no file operands are specified, or ‘-’ is specified, then the standard input will be used.

EXAMPLES
EXAMPLE 1 Examples of the asa command.

The command

date | asa | lp

converts output from a .out to conform with conventional printers and directs it through a pipe to the printer.

The command

asa output

shows the contents of file output on a terminal as it would appear on a printer.

The following program is used in the next two examples:
EXAMPLE 1 Examples of the asa command. (Continued)

write(*,'(" Blank")')
write(*,'("0Zero ")')
write(*,'("+ Plus ")')
write(*,'("1One ")')
end

Example 1. With actual files:
a.out > MyOutputFile
asa < MyOutputFile | lp

Example 2. With only pipes:
a.out | asa | lp

Both of the above examples produce two pages of output:

Page 1:
Blank
ZeroPlus

Page 2:
One

ENVIRONMENT VARIABLES
See environ(5) for descriptions of the following environment variables that affect the execution of asa: LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS
The following exit values are returned:

0 All input files were output successfully.
>0 An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO
lp(1), attributes(5), environ(5)
NAME

at, batch – execute commands at a later time

SYNOPSIS

at [-c | -k | -s] [-m] [-f file] [-p project] [-q queue name] -t time
at [-c | -k | -s] [-m] [-f file] [-p project] [-q queue name] timespec...
batch [-p project]

at

The at utility reads commands from standard input and groups them together as an at-job, to be executed at a later time.

The at-job will be executed in a separate invocation of the shell, running in a separate process group with no controlling terminal, except that the environment variables, current working directory, file creation mask (see umask(1)), and system resource limits (for sh and ksh only, see ulimit(1)) in effect when the at utility is executed will be retained and used when the at-job is executed.

When the at-job is submitted, the at_job_id and scheduled time are written to standard error. The at_job_id is an identifier that will be a string consisting solely of alphanumeric characters and the period character. The at_job_id is assigned by the system when the job is scheduled such that it uniquely identifies a particular job.

User notification and the processing of the job's standard output and standard error are described under the -m option.

Users are permitted to use at and batch (see below) if their name appears in the file /usr/lib/cron/at.allow. If that file does not exist, the file /usr/lib/cron/at.deny is checked to determine if the user should be denied access to at. If neither file exists, only a user with the solaris.jobs.user authorization is allowed to submit a job. If only at.deny exists and is empty, global usage is permitted. The at.allow and at.deny files consist of one user name per line.

cron and at jobs will be not be executed if the user's account is locked. Only accounts which are not locked as defined in shadow(4) will have their job or process executed.

batch

The batch utility reads commands to be executed at a later time. It is the equivalent of the command:

at -q b -m now

where queue b is a special at queue, specifically for batch jobs. Batch jobs will be submitted to the batch queue for immediate execution.

OPTIONS

The following options are supported. If the -c, -k, or -s options are not specified, the SHELL environment variable by default determines which shell to use.

- c C shell. csh(1) is used to execute the at-job.
- k Korn shell. ksh(1) is used to execute the at-job.
**at(1)**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-s</code></td>
<td>Bourne shell. <code>sh(1)</code> is used to execute the at-job.</td>
</tr>
<tr>
<td><code>-f file</code></td>
<td>Specifies the path of a file to be used as the source of the at-job, instead of standard input.</td>
</tr>
<tr>
<td><code>-l</code></td>
<td>(The letter ell.) Reports all jobs scheduled for the invoking user if no <code>at_job_id</code> operands are specified. If <code>at_job_ids</code> are specified, reports only information for these jobs.</td>
</tr>
<tr>
<td><code>-m</code></td>
<td>Sends mail to the invoking user after the at-job has run, announcing its completion. Standard output and standard error produced by the at-job will be mailed to the user as well, unless redirected elsewhere. Mail will be sent even if the job produces no output. If <code>-m</code> is not used, the job's standard output and standard error will be provided to the user by means of mail, unless they are redirected elsewhere; if there is no such output to provide, the user is not notified of the job's completion.</td>
</tr>
<tr>
<td><code>-p project</code></td>
<td>Specifies under which project the at or batch job will be run. When used with the <code>-l</code> option, limits the search to that particular project. Values for <code>project</code> will be interpreted first as a project name, and then as a possible project ID, if entirely numeric. By default, the user's current project is used.</td>
</tr>
<tr>
<td><code>-q queuename</code></td>
<td>Specifies in which queue to schedule a job for submission. When used with the <code>-l</code> option, limits the search to that particular queue. Values for <code>queuename</code> are limited to the lower case letters a through z. By default, at-jobs will be scheduled in queue a. In contrast, queue b is reserved for batch jobs. Since queue c is reserved for cron jobs, it can not be used with the <code>-q</code> option.</td>
</tr>
<tr>
<td><code>-r at_job_id</code></td>
<td>Removes the jobs with the specified <code>at_job_id</code> operands that were previously scheduled by the <code>at</code> utility.</td>
</tr>
<tr>
<td><code>-t time</code></td>
<td>Submits the job to be run at the time specified by the <code>time</code> option-argument, which must have the format as specified by the <code>touch(1)</code> utility.</td>
</tr>
</tbody>
</table>

**OPERANDS**

The following operands are supported:

- `at_job_id` The name reported by a previous invocation of the `at` utility at the time the job was scheduled.
- `timespec` Submit the job to be run at the date and time specified. All of the `timespec` operands are interpreted as if they were separated by space characters and concatenated. The date and time are interpreted as being in the timezone of the user (as determined by the `TZ` variable), unless a timezone name appears as part of `time` below.
In the "C" locale, the following describes the three parts of the time specification string. All of the values from the LC_TIME categories in the "C" locale are recognized in a case-insensitive manner.

**time**

The time can be specified as one, two or four digits. One- and two-digit numbers are taken to be hours, four-digit numbers to be hours and minutes. The time can alternatively be specified as two numbers separated by a colon, meaning *hour:minute*. An AM/PM indication (one of the values from the am_pm keywords in the LC_TIME locale category) can follow the time; otherwise, a 24-hour clock time is understood. A timezone name of GMT, UCT, or ZULU (case insensitive) can follow to specify that the time is in Coordinated Universal Time. Other timezones can be specified using the TZ environment variable. The time field can also be one of the following tokens in the "C" locale:

- **midnight** Indicates the time 12:00 am (00:00).
- **noon** Indicates the time 12:00 pm.
- **now** Indicate the current day and time. Invoking `at now` will submit an at-job for potentially immediate execution (that is, subject only to unspecified scheduling delays).

**date**

An optional date can be specified as either a month name (one of the values from the mon or abmon keywords in the LC_TIME locale category) followed by a day number (and possibly year number preceded by a comma) or a day of the week (one of the values from the day or abday keywords in the LC_TIME locale category). Two special days are recognized in the "C" locale:

- **today** Indicates the current day.
- **tomorrow** Indicates the day following the current day.

If no date is given, today is assumed if the given time is greater than the current time, and tomorrow is assumed if it is less. If the given month is less than the current month (and no year is given), next year is assumed.
The optional *increment* is a number preceded by a plus sign (+) and suffixed by one of the following: minutes, hours, days, weeks, months, or years. (The singular forms will be also accepted.) The keyword *next* is equivalent to an increment number of + 1. For example, the following are equivalent commands:

```
at 2pm + 1 week
at 2pm next week
```

**USAGE**

The format of the `at` command line shown here is guaranteed only for the "C" locale. Other locales are not supported for midnight, noon, now, mon, abmon, day, abday, today, tomorrow, minutes, hours, days, weeks, months, years, and next.

Since the commands run in a separate shell invocation, running in a separate process group with no controlling terminal, open file descriptors, traps and priority inherited from the invoking environment are lost.

**EXAMPLE 1** Typical sequence at a terminal

This sequence can be used at a terminal:

```
$ at -m 0730 tomorrow
sort < file >outfile
<EOT>
```

**EXAMPLE 2** Redirecting output

This sequence, which demonstrates redirecting standard error to a pipe, is useful in a command procedure (the sequence of output redirection specifications is significant):

```
$ at now + 1 hour <<!
diff file1 file2 2>&1 >outfile | mailx mygroup
```

**EXAMPLE 3** Self-rescheduling a job

To have a job reschedule itself, `at` can be invoked from within the at-job. For example, this "daily-processing" script named `my.daily` will run every day (although `crontab` is a more appropriate vehicle for such work):

```
# my.daily runs every day
at now tomorrow < my.daily
daily-processing
```
EXAMPLE 3 Self-rescheduling a job  (Continued)

EXAMPLE 4 Various time and operand presentations

The spacing of the three portions of the "C" locale timespec is quite flexible as long as there are no ambiguities. Examples of various times and operand presentations include:

at 0815am Jan 24
at 8:15am Jan 24
at now "+1 day"
at 5 pm Fri day
at '17 utc+'30 minutes'

EXAMPLE 5 Typical sequence at a terminal

This sequence can be used at a terminal:

$ batch
sort <file >outfile
<EOT>

EXAMPLE 6 Redirecting output

This sequence, which demonstrates redirecting standard error to a pipe, is useful in a command procedure (the sequence of output redirection specifications is significant):

$ batch <<!
diff file1 file2 2>&1 >outfile | mailx mygroup
!

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of at and batch: LC_CTYPE, LC_MESSAGES, NLSPATH, and LC_TIME.

SHELL

Determine a name of a command interpreter to be used to invoke the at-job. If the variable is unset or NULL, sh will be used. If it is set to a value other than sh, the implementation will use that shell; a warning diagnostic will be printed telling which shell will be used.

TZ

Determine the timezone. The job will be submitted for execution at the time specified by timespec or -t time relative to the timezone specified by the TZ variable. If timespec specifies a timezone, it will override TZ. If timespec does not specify a timezone and TZ is unset or NULL, an unspecified default timezone will be used.

DATEMSK

If the environment variable DATEMSK is set, at will use its value as the full path name of a template file containing format strings. The strings consist of format specifiers and text characters that are used to provide a richer set of allowable date formats in different
languages by appropriate settings of the environment variable LANG or LC_TIME. The list of allowable format specifiers is located in the getdate(3C) manual page. The formats described in the OPERANDS section for the time and date arguments, the special names noon, midnight, now, next, today, tomorrow, and the increment argument are not recognized when DATETIME is set.

EXIT STATUS

The following exit values are returned:

0 The at utility successfully submitted, removed or listed a job or jobs.

>0 An error occurred, and the job will not be scheduled.

FILES

/usr/lib/cron/at.allow names of users, one per line, who are authorized access to the at and batch utilities

/usr/lib/cron/at.deny names of users, one per line, who are denied access to the at and batch utilities

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>Not enabled</td>
</tr>
</tbody>
</table>

at

SEE ALSO

auths(1), crontab(1), csh(1), date(1), ksh(1), sh(1), touch(1), ulimit(1), umask(1), cron(1M), getdate(3C), auth_attr(4), shadow(4), attributes(5), environ(5)

NOTES

Regardless of queue used, cron(1M) has a limit of 100 jobs in execution at any time.

There can be delays in cron at job execution. In some cases, these delays can compound to the point that cron job processing appears to be hung. All jobs will be executed eventually. When the delays are excessive, the only workaround is to kill and restart cron.
NAME  atq – display the jobs queued to run at specified times

SYNOPSIS  atq [-c] [-n] [username...]

DESCRIPTION  The atq utility displays the at jobs queued up for the current user. at(1) is a utility that allows users to execute commands at a later date. If invoked by a user with the solaris.jobs.admin authorization, atq will display all jobs in the queue.

If no options are given, the jobs are displayed in chronological order of execution.

When an authorized user invokes atq without specifying username, the entire queue is displayed; when a username is specified, only those jobs belonging to the named user are displayed.

OPTIONS  The following options are supported:

- c  Displays the queued jobs in the order they were created (that is, the time that the at command was given).

- n  Displays only the total number of jobs currently in the queue.

FILES  /var/spool/cron/atjobs  spool area for at jobs.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO  at(1), atrm(1), auths(1), cron(1M), auth_attr(4), attributes(5)
The `atrm` utility removes delayed-execution jobs that were created with the `at(1)` command, but have not yet executed. The list of these jobs and associated job numbers can be displayed by using `atq(1)`.

`atrm` removes each job-number you specify, and/or all jobs belonging to the user you specify, provided that you own the indicated jobs.

You can only remove jobs belonging to other users if you have `solaris.jobs.admin` privileges.

The following options are supported:

- `-a` All. Removes all unexecuted jobs that were created by the current user. If invoked by the privileged user, the entire queue will be flushed.
- `-f` Force. All information regarding the removal of the specified jobs is suppressed.
- `-i` Interactive. `atrm` asks if a job should be removed. If you respond with a `y`, the job will be removed.

`/var/spool/cron/atjobs` spool area for at jobs

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO `at(1), atq(1), auths(1), cron(1M), auth_attr(4), attributes(5)`
audioconvert(1)

NAME  audioconvert – convert audio file formats

SYNOPSIS  audioconvert [-pF] [-f outfmt] [-o outfile] [ [-i infmt] [file...]] ...

DESCRIPTION  audioconvert converts audio data between a set of supported audio encodings and file formats. It can be used to compress and decompress audio data, to add audio file headers to raw audio data files, and to convert between standard data encodings, such as -law and linear PCM.

If no filenames are present, audioconvert reads the data from the standard input stream and writes an audio file to the standard output. Otherwise, input files are processed in order, concatenated, and written to the output file.

Input files are expected to contain audio file headers that identify the audio data format. If the audio data does not contain a recognizable header, the format must be specified with the -i option, using the rate, encoding, and channels keywords to identify the input data format.

The output file format is derived by updating the format of the first input file with the format options in the -f specification. If -p is not specified, all subsequent input files are converted to this resulting format and concatenated together. The output file will contain an audio file header, unless format=raw is specified in the output format options.

Input files may be converted in place by using the -p option. When -p is in effect, the format of each input file is modified according to the -f option to determine the output format. The existing files are then overwritten with the converted data.

The file(1) command decodes and prints the audio data format of Sun audio files.

OPTIONS  The following options are supported:

-p  In Place: The input files are individually converted to the format specified by the -f option and rewritten. If a target file is a symbolic link, the underlying file will be rewritten. The -o option may not be specified with -p.

-F  Force: This option forces audioconvert to ignore any file header for input files whose format is specified by the -i option. If -F is not specified, audioconvert ignores the -i option for input files that contain valid audio file headers.

-f outfmt  Output Format: This option is used to specify the file format and data encoding of the output file. Defaults for unspecified fields are derived from the input file format. Valid keywords and values are listed in the next section.

-o outfile  Output File: All input files are concatenated, converted to the output format, and written to the named output file. If -o and -p are not specified, the concatenated output is written to the standard output. The -p option may not be specified with -o.
**Input Format**: This option is used to specify the data encoding of raw input files. Ordinarily, the input data format is derived from the audio file header. This option is required when converting audio data that is not preceded by a valid audio file header. If `-i` is specified for an input file that contains an audio file header, the input format string will be ignored, unless `-F` is present. The format specification syntax is the same as the `-f` output file format.

Multiple input formats may be specified. An input format describes all input files following that specification, until a new input format is specified.

**File Specification**: The named audio files are concatenated, converted to the output format, and written out. If no file name is present, or if the special file name `−` is specified, audio data is read from the standard input.

**Help**: Prints a command line usage message.

The syntax for the input and output format specification is:

```
keyword=value[keyword=value ... ]
```

with no intervening whitespace. Unambiguous values may be used without the preceding `keyword=`,

**Rate**: The audio sampling rate is specified in samples per second. If a number is followed by the letter `k`, it is multiplied by 1000 (for example, 44.1k = 44100). Standard of the commonly used sample rates are: 8k, 16k, 32k, 44.1k, and 48k.

**Channels**: The number of interleaved channels is specified as an integer. The words `mono` and `stereo` may also be used to specify one and two channel data, respectively.

**Encoding**: This option specifies the digital audio data representation. Encodings determine precision implicitly (`ulaw` implies 8-bit precision) or explicitly as part of the name (for example, `linear16`). Valid encoding values are:

- `ulaw` CCITT G.711 -law encoding. This is an 8-bit format primarily used for telephone quality speech.

- `alaw` CCITT G.711 A-law encoding. This is an 8-bit format primarily used for telephone quality speech in Europe.

- `linear8`,

- `linear16`,

User Commands 75
**audioconvert(1)**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>linear32</td>
<td>Linear Pulse Code Modulation (PCM) encoding. The name identifies the number of bits of precision. linear16 is typically used for high quality audio data.</td>
</tr>
<tr>
<td>pcm</td>
<td>Same as linear16.</td>
</tr>
<tr>
<td>g721</td>
<td>CCITT G.721 compression format. This encoding uses Adaptive Delta Pulse Code Modulation (ADPCM) with 4-bit precision. It is primarily used for compressing -law voice data (achieving a 2:1 compression ratio).</td>
</tr>
<tr>
<td>g723</td>
<td>CCITT G.723 compression format. This encoding uses Adaptive Delta Pulse Code Modulation (ADPCM) with 3-bit precision. It is primarily used for compressing -law voice data (achieving an 8:3 compression ratio). The audio quality is similar to G.721, but may result in lower quality when used for non-speech data.</td>
</tr>
</tbody>
</table>

The following encoding values are also accepted as shorthand to set the sample rate, channels, and encoding:

- **voice** Equivalent to `encoding=ulaw,rate=8k,channels=mono`.
- **cd** Equivalent to `encoding=linear16,rate=44.1k,channels=stereo`.
- **dat** Equivalent to `encoding=linear16,rate=48k,channels=stereo`.

**format** This option specifies the audio file format. Valid formats are:

- **sun** Sun compatible file format (the default).
- **raw** Use this format when reading or writing raw audio data (with no audio header), or in conjunction with an offset to import a foreign audio file format.

**offset** `(-i only)` Specifies a byte offset to locate the start of the audio data. This option may be used to import audio data that contains an unrecognized file header.

**USAGE** See largefile(5) for the description of the behavior of audioconvert when encountering files greater than or equal to 2 Gbyte (2³¹ bytes).

**EXAMPLES**

**EXAMPLE 1** Recording and compressing voice data before storing it

Record voice data and compress it before storing it to a file:

```bash
example% audiorecord | audioconvert -f g721 > mydata.au
```
EXAMPLE 2 Concatenating two audio files

Concatenate two Sun format audio files, regardless of their data format, and output an 8-bit ulaw, 16 kHz, mono file:

```
example% audioconvert -f ulaw,rate=16k,mono -o outfile.au infile1 infile2
```

EXAMPLE 3 Converting a directory to Sun format

Convert a directory containing raw voice data files, in place, to Sun format (adds a file header to each file):

```
example% audioconvert -p -i voice -f sun *.au
```

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>SPARC, IA</td>
</tr>
<tr>
<td>Availability</td>
<td>SUNWauda</td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Evolving</td>
</tr>
</tbody>
</table>

SEE ALSO

audioplay(1), audierecord(1), file(1), attributes(5), largefile(5)

NOTES

The algorithm used for converting multi-channel data to mono is implemented by simply summing the channels together. If the input data is perfectly in phase (as would be the case if a mono file is converted to stereo and back to mono), the resulting data may contain some distortion.
The audioplay utility copies the named audio files (or the standard input if no filenames are present) to the audio device. If no input file is specified and standard input is a tty, the port, volume, and balance settings specified on the command line will be applied and the program will exit.

The input files must contain a valid audio file header. The encoding information in this header is matched against the capabilities of the audio device and, if the data formats are incompatible, an error message is printed and the file is skipped. Compressed ADPCM (G.721) monaural audio data is automatically uncompressed before playing.

Minor deviations in sampling frequency (that is, less than 1%) are ordinarily ignored. This allows, for instance, data sampled at 8012 Hz to be played on an audio device that only supports 8000 Hz. If the -V option is present, such deviations are flagged with warning messages.

The following options are supported:

- `-i`
  **Immediate**: If the audio device is unavailable (that is, another process currently has write access), audioplay ordinarily waits until it can obtain access to the device. When the `-i` option is present, audioplay prints an error message and exits immediately if the device is busy.

- `-V`
  **Verbose**: Prints messages on the standard error when waiting for access to the audio device or when sample rate deviations are detected.

- `-v vol`
  **Volume**: The output volume is set to the specified value before playing begins, and is reset to its previous level when audioplay exits. The `vol` argument is an integer value between 0 and 100, inclusive. If this argument is not specified, the output volume remains at the level most recently set by any process.

- `-b bal`
  **Balance**: The output balance is set to the specified value before playing begins, and is reset to its previous level when audioplay exits. The `bal` argument is an integer value between -100 and 100, inclusive. A value of -100 indicates left balance, 0 middle, and 100 right. If this argument is not specified, the output balance remains at the level most recently set by any process.

- `-p speaker | headphone | line`
  **Output Port**: Selects the built-in speaker (the default), headphone jack, or line out as the destination of the audio output signal. If this argument is not specified, the output port will remain unchanged. **Please note**: Not all audio adapters support all of the output ports. If the named port does not exist, an appropriate substitute will be used.
-\  ?
  Help: Prints a command line usage message.

OPERANDS
  file  File Specification: Audio files named on the command line are played sequentially. If no filenames are present, the standard input stream (if it is not a tty) is played (it, too, must contain an audio file header). The special filename ‘−’ may be used to read the standard input stream instead of a file. If a relative path name is supplied, the AUDIOPATH environment variable is consulted (see below).

USAGE
  See largefile(5) for the description of the behavior of audioplay when encountering files greater than or equal to 2 Gbyte (2^31 bytes).

ENVIRONMENT VARIABLES
  AUDIODEV  The full path name of the audio device to write to, if no -d argument is supplied. If the AUDIODEV variable is not set, /dev/audio is used.
  AUDIOPATH  A colon-separated list of directories in which to search for audio files whose names are given by relative pathnames. The current directory (".") may be specified explicitly in the search path. If the AUDIOPATH variable is not set, only the current directory will be searched.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>SPARC, IA</td>
</tr>
<tr>
<td>Availability</td>
<td>SUNWauda</td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Evolving</td>
</tr>
</tbody>
</table>

SEE ALSO
  audioconvert(1), audiorecord(1), mixerctl(1), attributes(5), largefile(5), usb_ac(7D), audio(7I), mixer(7I)

BUGS
  audioplay currently supports a limited set of audio format conversions. If the audio file is not in a format supported by the audio device, it must first be converted. For example, to convert to voice format on the fly, use the command:

  example\ audioconvert -f voice myfile | audioplay

  The format conversion will not always be able to keep up with the audio output. If this is the case, you should convert to a temporary file before playing the data.
NAME
audiorecord – record an audio file

SYNOPSIS

DESCRIPTION
The audiorecord utility copies audio data from the audio device to a named audio file (or the standard output if no filename is present). If no output file is specified and standard output is a tty, the volume, balance, monitor volume, port, and audio format settings specified on the command line will be applied and the program will exit.

By default, monaural audio data is recorded at 8 kHz and encoded in -law format. If the audio device supports additional configurations, the -c, -s, and -e options may be used to specify the data format. The output file is prefixed by an audio file header that identifies the format of the data encoded in the file.

Recording begins immediately and continues until a SIGINT signal (for example, Ctrl-C) is received. If the -t option is specified, audiorecord stops when the specified quantity of data has been recorded.

If the audio device is unavailable (that is, another process currently has read access), audiorecord prints an error message and exits immediately.

OPTIONS
The following options are supported:

-a
Append: Appends the data on the end of the named audio file. The audio device must support the audio data format of the existing file.

-f
Force: When the -a flag is specified, the sample rate of the audio device must match the sample rate at which the original file was recorded. If the -f flag is also specified, sample rate differences are ignored, with a warning message printed on the standard error.

-v vol
Volume: The recording gain is set to the specified value before recording begins, and is reset to its previous level when audiorecord exits. The vol argument is an integer value between 0 and 100, inclusive. If this argument is not specified, the input volume will remain at the level most recently set by any process.

-b bal
Balance: The recording balance is set to the specified value before recording begins, and is reset to its previous level when audiorecord exits. The bal argument is an integer value between -100 and 100, inclusive. A value of 0 indicates left balance, 0 middle, and 100 right. If this argument is not specified, the input balance will remain at the level most recently set by any process.

-m monvol
Monitor Volume: The input monitor volume is set to the specified value before recording begins, and is reset to its previous level when audiorecord exits. The monvol argument is an integer value between 0 and 100, inclusive. A non-zero value
allows a directly connected input source to be heard on the output speaker while recording is in-progress. If this argument is not specified, the monitor volume will remain at the level most recently set by any process.

-\p mic | line | internal-cd
  Input Port: Selects the mic, line, or internal-cd input as the source of the audio output signal. If this argument is not specified, the input port will remain unchanged. Please note: Some systems will not support all possible input ports. If the named port does not exist, this option is ignored.

-\c channels
  Channels: Specifies the number of audio channels (1 or 2). The value may be specified as an integer or as the string mono or stereo. The default value is mono.

-\s rate
  Sample Rate: Specifies the sample rate, in samples per second. If a number is followed by the letter k, it is multiplied by 1000 (for example, 44.1k = 44100). The default sample rate is 8 kHz.

-\e encoding
  Encoding: Specifies the audio data encoding. This value may be one of ulaw, alaw, or linear. The default encoding is ulaw.

-\t time
  Time: The time argument specifies the maximum length of time to record. Time can be specified as a floating-point value, indicating the number of seconds, or in the form: hh:mm:ss.dd, where the hour and minute specifications are optional.

-\i info
  Information: The ‘information’ field of the output file header is set to the string specified by the info argument. This option cannot be specified in conjunction with the -a argument.

-\d dev
  Device: The dev argument specifies an alternate audio device from which input should be taken. If the -d option is not specified, the AUDIODEV environment variable is consulted (see below). Otherwise, /dev/audio is used as the default audio device.

-\?
  Help: Prints a command line usage message.

**OPERANDS**

file
  File Specification: The named audio file is rewritten (or appended). If no filename is present (and standard output is not a tty), or if the special filename ‘−’ is specified, output is directed to the the standard output.

**USAGE**

See largefile(5) for the description of the behavior of audiorecord when encountering files greater than or equal to 2 Gbyte (2³¹ bytes).

**ENVIRONMENT VARIABLES**

AUDIODEV
  The full path name of the audio device to record from, if no -d argument is supplied. If the AUDIODEV variable is not set, /dev/audio is used.
See attributes(5) for descriptions of the following attributes:

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>SPARC, IA</td>
</tr>
<tr>
<td>Availability</td>
<td>SUNWauda</td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Evolving</td>
</tr>
</tbody>
</table>

SEE ALSO
audioconvert(1), audioplay(1), mixerctl(1), attributes(5), largefile(5), usb_ac(7D), audio(7I), mixer(7I)
### NAME
auths – print authorizations granted to a user

### SYNOPSIS
```
auths [ user ...]
```

### DESCRIPTION
The `auths` command prints on standard output the authorizations that you or the optionally-specified user or role have been granted. Authorizations are rights that are checked by certain privileged programs to determine whether a user may execute restricted functionality.

Each user may have zero or more authorizations. Authorizations are represented by fully-qualified names, which identify the organization that created the authorization and the functionality that it controls. Following the Java convention, the hierarchical components of an authorization are separated by dots (.), starting with the reverse order Internet domain name of the creating organization, and ending with the specific function within a class of authorizations.

An asterisk (*) indicates all authorizations in a class.

A user’s authorizations are looked up in `user_attr(4)` and in the `/etc/security/policy.conf` file (see `policy.conf(4)`). Authorizations may be specified directly in `user_attr(4)` or indirectly through `prof_attr(4)`. Authorizations may also be assigned to every user in the system directly as default authorizations or indirectly as default profiles in the `/etc/security/policy.conf` file.

### EXAMPLES
#### EXAMPLE 1 Sample output
The `auths` output has the following form:
```
example% auths tester01 tester02
tester01 : com.sun.system.date, com.sun.jobs.admin
tester02 : com.sun.system.*
example%
```

### EXIT STATUS
The following exit values are returned:
- **0**  Successful completion.
- **1**  An error occurred.

### FILES
- `/etc/user_attr`
- `/etc/security/auth_attr`
- `/etc/security/policy.conf`
- `/etc/security/prof_attr`

### ATTRIBUTES
See `attributes(5)` for descriptions of the following attributes:
auths(1)

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO profiles(1), roles(1), getauthattr(3SECDB), auth_attr(4), policy.conf(4), prof_attr(4), user_attr(4), attributes(5)
awk – pattern scanning and processing language

SYNOPSIS

/usr/bin/awk [-f progfile] [-F c] [' prog ' ] [parameters] [filename...]
/usr/xpg4/bin/awk [-F ERE] [-v assignment...] 'program' -f progfile...

DESCRIPTION

The /usr/xpg4/bin/awk utility is described on the awk(1) manual page.

The /usr/bin/awk utility scans each input filename for lines that match any of a set of patterns specified in prog. The prog string must be enclosed in single quotes ( ´ ) to protect it from the shell. For each pattern in prog there may be an associated action performed when a line of a filename matches the pattern. The set of pattern-action statements may appear literally as prog or in a file specified with the -f progfile option. Input files are read in order; if there are no files, the standard input is read. The filename ‘-’ means the standard input.

OPTIONS

The following options are supported:

- f progfile awk uses the set of patterns it reads from progfile.
- Fc Uses the character c as the field separator (FS) character. See the discussion of FS below.

Input Lines

Each input line is matched against the pattern portion of every pattern-action statement; the associated action is performed for each matched pattern. Any filename of the form var=value is treated as an assignment, not a filename, and is executed at the time it would have been opened if it were a filename. Variables assigned in this manner are not available inside a BEGIN rule, and are assigned after previously specified files have been read.

An input line is normally made up of fields separated by white spaces. (This default can be changed by using the FS built-in variable or the -Fc option.) The default is to ignore leading blanks and to separate fields by blanks and/or tab characters. However, if FS is assigned a value that does not include any of the white spaces, then leading blanks are not ignored. The fields are denoted $1, $2, ... ; $0 refers to the entire line.

Pattern-action Statements

A pattern-action statement has the form:

pattern { action }

Either pattern or action may be omitted. If there is no action, the matching line is printed. If there is no pattern, the action is performed on every input line.

Pattern-action statements are separated by newlines or semicolons.

Patterns are arbitrary Boolean combinations ( !, ||, &&, and parentheses) of relational expressions and regular expressions. A relational expression is one of the following:

expression relop expression
expression matchop regular_expression
where a relop is any of the six relational operators in C, and a matchop is either ~ (contains) or !~ (does not contain). An expression is an arithmetic expression, a relational expression, the special expression

`var in array`

or a Boolean combination of these.

Regular expressions are as in egrep(1). In patterns they must be surrounded by slashes. Isolated regular expressions in a pattern apply to the entire line. Regular expressions may also occur in relational expressions. A pattern may consist of two patterns separated by a comma; in this case, the action is performed for all lines between the occurrence of the first pattern to the occurrence of the second pattern.

The special patterns BEGIN and END may be used to capture control before the first input line has been read and after the last input line has been read respectively. These keywords do not combine with any other patterns.

### Built-in Variables

Built-in variables include:

- **FILENAME** - name of the current input file
- **FS** - input field separator regular expression (default blank and tab)
- **NF** - number of fields in the current record
- **NR** - ordinal number of the current record
- **OFMT** - output format for numbers (default `% .6g`)
- **OFS** - output field separator (default blank)
- **ORS** - output record separator (default new-line)
- **RS** - input record separator (default new-line)

An action is a sequence of statements. A statement may be one of the following:

- `if ( expression ) statement [ else statement ]`
- `while ( expression ) statement`
- `do statement while ( expression )`
- `for ( expression ; expression ; expression ) statement`
- `for ( var in array ) statement`
- `break`
- `continue`
- `{ [ statement ] . . . }

expression  # commonly variable = expression
print [ expression-list ] [ >expression ]
printf format [ ,expression-list ] [ >expression ]
next  # skip remaining patterns on this input line
exit [ expr ]  # skip the rest of the input; exit status is expr
Statements are terminated by semicolons, newlines, or right braces. An empty expression-list stands for the whole input line. Expressions take on string or numeric values as appropriate, and are built using the operators +, −, *, /, %, ^ and concatenation (indicated by a blank). The operators ++, −−, +=, −=, *=, /=, %=, ^=, >, >=, <, <=, ==, !=, and ?:: are also available in expressions. Variables may be scalars, array elements (denoted x[i]), or fields. Variables are initialized to the null string or zero. Array subscripts may be any string, not necessarily numeric; this allows for a form of associative memory. String constants are quoted (""), with the usual C escapes recognized within.

The print statement prints its arguments on the standard output, or on a file if >expression is present, or on a pipe if ’cmd’ is present. The output resulted from the print statement is terminated by the output record separator with each argument separated by the current output field separator. The printf statement formats its expression list according to the format (see printf(3C)).

The arithmetic functions are as follows:

\begin{itemize}
  \item \texttt{cos(x)} \quad \text{Return cosine of x, where x is in radians. (In /usr/xpg4/bin/awk only. See awk(1).)}
  \item \texttt{sin(x)} \quad \text{Return sine of x, where x is in radians. (In /usr/xpg4/bin/awk only. See awk(1).)}
  \item \texttt{exp(x)} \quad \text{Return the exponential function of x.}
  \item \texttt{log(x)} \quad \text{Return the natural logarithm of x.}
  \item \texttt{sqrt(x)} \quad \text{Return the square root of x.}
  \item \texttt{int(x)} \quad \text{Truncate its argument to an integer. It will be truncated toward 0 when x > 0.}
\end{itemize}

The string functions are as follows:

\begin{itemize}
  \item \texttt{index(s, t)} \quad \text{Return the position in string s where string t first occurs, or 0 if it does not occur at all.}
  \item \texttt{int(s)} \quad \text{truncates s to an integer value. If s is not specified, $0$ is used.}
  \item \texttt{length(s)} \quad \text{Return the length of its argument taken as a string, or of the whole line if there is no argument.}
  \item \texttt{split(s, a, fs)} \quad \text{Split the string s into array elements a[1], a[2], \ldots a[n], and returns n. The separation is done with the regular expression \texttt{fs} or with the field separator \texttt{FS} if \texttt{fs} is not given.}
\end{itemize}
**awk(1)**

`sprintf(fmt, expr, expr, ...)`

Format the expressions according to the `printf(3C)` format given by `fmt` and returns the resulting string.

`substr(s, m, n)`

returns the `n`-character substring of `s` that begins at position `m`.

The input/output function is as follows:

`getline` Set `$0` to the next input record from the current input file.

`getline` returns `1` for successful input, `0` for end of file, and `-1` for an error.

**Large File Behavior**

See `largefile(5)` for the description of the behavior of `awk` when encountering files greater than or equal to 2 Gbyte (2^31 bytes).

**EXAMPLES**

**EXAMPLE 1** Print lines longer than 72 characters:

```awk
length > 72
```

**EXAMPLE 2** Print first two fields in opposite order:

```awk
{ print $2, $1 }
```

**EXAMPLE 3** Same, with input fields separated by comma and/or blanks and tabs:

```awk
BEGIN { FS = "*, [\t]*|[\t]+" }
{ print $2, $1 }
```

**EXAMPLE 4** Add up first column, print sum and average:

```awk
{ s += $1 }
END { print "sum is", s, " average is", s/NR }
```

**EXAMPLE 5** Printing fields in reverse order

```awk
{ for (i = NF; i > 0; −−i) print $i }
```

**EXAMPLE 6** Print all lines between start/stop pairs:

```awk
/start/, /stop/
```

**EXAMPLE 7** Print all lines whose first field is different from the previous one:

```awk
$1 != prev { print; prev = $1 }
```

**EXAMPLE 8** Print a file, filling in page numbers starting at 5:

```awk
/Page/ { $2 = n++; }
{ print }
```
EXAMPLE 8  Print a file, filling in page numbers starting at 5:  (Continued)

EXAMPLE 9  Print a file and number its pages starting at 5:

Assuming this program is in a file named prog, the following command line prints
the file input numbering its pages starting at 5:

awk f prog n=5 input

ENVIRONMENT VARIABLES  See environ(5) for descriptions of the following environment variables that affect the
execution of awk: LC_CTYPE and LC_MESSAGES.

LC_NUMERIC   Determine the radix character used when interpreting numeric
input, performing conversions between numeric and string values
and formatting numeric output. Regardless of locale, the period
character (the decimal-point character of the POSIX locale) is the
decimal-point character recognized in processing awk programs
(including assignments in command-line arguments).

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

<table>
<thead>
<tr>
<th>/usr/bin/awk</th>
<th>/usr/xpg4/bin/awk</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTRIBUTE TYPE</td>
<td>ATTRIBUTE TYPE</td>
</tr>
<tr>
<td>Availability</td>
<td>Availability</td>
</tr>
<tr>
<td>CSI</td>
<td>CSI</td>
</tr>
<tr>
<td>Attribute Value</td>
<td>Attribute Value</td>
</tr>
<tr>
<td>SUNWesu</td>
<td>SUNWesu</td>
</tr>
<tr>
<td>Enabled</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

SEE ALSO  egrep(1), grep(1), nawk(1), sed(1), printf(3C), attributes(5), environ(5),
largefile(5), XPG4(5)

NOTES  Input white space is not preserved on output if fields are involved.

There are no explicit conversions between numbers and strings. To force an expression
to be treated as a number add 0 to it; to force it to be treated as a string concatenate the
null string (""") to it.
banner(1)

**NAME**  banner – make posters

**SYNOPSIS**  banner strings

**DESCRIPTION**  banner prints its arguments (each up to 10 characters long) in large letters on the standard output.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
</tbody>
</table>

**SEE ALSO**  echo(1), attributes(5)
NAME
basename, dirname – deliver portions of path names

SYNOPSIS
/usr/bin/basename string [suffix]
/usr/xpg4/bin/basename string [suffix]
dirname string

DESCRIPTION
The basename utility deletes any prefix ending in / and the suffix (if present in string) from string, and prints the result on the standard output. It is normally used inside substitution marks (‘ ‘) within shell procedures.

/usr/bin
The suffix is a pattern defined on the expr(1) manual page.

/usr/xpg4/bin
The suffix is a string with no special significance attached to any of the characters it contains.

The dirname utility delivers all but the last level of the path name in string.

EXAMPLES
EXAMPLE 1 Setting environment variables
The following example, invoked with the argument /home/sms/personal/mail sets the environment variable NAME to the file named mail and the environment variable MYMAILPATH to the string /home/sms/personal:

```
example% NAME='basename $HOME/personal/mail'
exaple% MYMAILPATH='dirname $HOME/personal/mail'
```

EXAMPLE 2 Compiling a file and moving the output
This shell procedure, invoked with the argument /usr/src/bin/cat.c, compiles the named file and moves the output to cat in the current directory:

```
example% cc $1
example% mv a.out `basename $1 .c`
```

ENVIRONMENT VARIABLES
See environ(5) for descriptions of the following environment variables that affect the execution of basename and dirname: LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS
The following exit values are returned:

- 0 Successful completion.
- >0 An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>
basename(1)

/usr/xpg4/bin

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWxcu4</td>
</tr>
</tbody>
</table>

SEE ALSO

expr(1), attributes(5), environ(5), XPG4(5)
NAME    basename – display portions of pathnames

SYNOPSIS /usr/ucb/basename string [suffix]

DESCRIPTION The basename utility deletes any prefix ending in ‘/’ and the suffix, if present in string. It directs the result to the standard output, and is normally used inside substitution marks (‘ ’) within shell procedures. The suffix is a string with no special significance attached to any of the characters it contains.

EXAMPLES EXAMPLE 1 Using the basename command.

This shell procedure invoked with the argument /usr/src/bin/cat.c compiles the named file and moves the output to cat in the current directory:

example% cc $1
example% mv a.out 'basename $1 .c'

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

SEE ALSO sh(1), attributes(5)
**NAME** | bc – arbitrary precision arithmetic language  
**SYNOPSIS** | `bc [-c] [-l] [file...]`  
**DESCRIPTION** | The `bc` utility implements an arbitrary precision calculator. It takes input from any files given, then reads from the standard input. If the standard input and standard output to `bc` are attached to a terminal, the invocation of `bc` is interactive, causing behavioral constraints described in the following sections. `bc` processes a language that resembles C and is a preprocessor for the desk calculator program `dc`, which it invokes automatically unless the `-c` option is specified. In this case the `dc` input is sent to the standard output instead.  
**USAGE** | The syntax for `bc` programs is as follows:  
- `L` means a letter a-z,  
- `E` means an expression: a (mathematical or logical) value, an operand that takes a value, or a combination of operands and operators that evaluates to a value,  
- `S` means a statement.  
**Comments** | Enclosed in /* and */.  
**Names (Operands)** | Simple variables: `L`. Array elements: `L [ E ]` (up to `BC_DIM_MAX` dimensions). The words `ibase`, `obase` (limited to `BC_BASE_MAX`), and `scale` (limited to `BC_SCALE_MAX`).  
**Other Operands** | Arbitrarily long numbers with optional sign and decimal point. Strings of fewer than `BC_STRING_MAX` characters, between double quotes (").  
`sqrt ( E )` Square root  
`length ( E )` Number of significant decimal digits.  
`scale ( E )` Number of digits right of decimal point.  
`L ( E , ..., E )`  
**Operators** | `+ − * / % ^`  
`( % is remainder; ^ is power)`  
`++ --`  
`(prefix and postfix; apply to names)`  
`== <= >= != < >`  
`= += -= *= /= %= ^=`  
**Statements** | `E`  
`{ S ;...; S }`  
`if ( E ) S`
while \((E)\) \(S\)
for \((E;E;E)\) \(S\)
nul statement
break
quit

/string

Function Definitions
\[
define L(L, \ldots, L)\
\text{auto } L, \ldots, L
S; \ldots S
\text{return (} E \text{)}
\]

Functions in -l Math Library
\[
s(x) \quad \text{sine}
c(x) \quad \text{cosine}
e(x) \quad \text{exponential}
l(x) \quad \text{log}
a(x) \quad \text{arctangent}
j(n,x) \quad \text{Bessel function}
\]

All function arguments are passed by value.

The value of a statement that is an expression is printed unless the main operator is an assignment. Either semicolons or new-lines may separate statements. Assignment to \textit{scale} influences the number of digits to be retained on arithmetic operations in the manner of \textit{dc}. Assignments to \textit{ibase} or \textit{obase} set the input and output number radix respectively.

The same letter may be used as an array, a function, and a simple variable simultaneously. All variables are global to the program. \textit{auto} variables are stacked during function calls. When using arrays as function arguments or defining them as automatic variables, empty square brackets must follow the array name.

OPTIONS
The following operands are supported:
\[
\text{-c} \quad \text{Compile only. The output is } \textit{dc} \text{ commands that are sent to the standard output.}
\text{-l} \quad \text{Define the math functions and initialize } \textit{scale} \text{ to 20, instead of the default zero.}
\]

OPERANDS
The following operands are supported:
\[
\text{file} \quad \text{A pathname of a text file containing } \textit{bc} \text{ program statements. After all cases of } \textit{file} \text{ have been read, } \textit{bc} \text{ will read the standard input.}
\]
EXAMPLE 1 Setting the precision of a variable

In the shell, the following assigns an approximation of the first ten digits of \( n \) to the variable \( x \):

\[ x = $(printf "%s\n" 'scale = 10; 104348/33215' | bc) \]

EXAMPLE 2 Defining a computing function

Defines a function to compute an approximate value of the exponential function:

```bash
scale = 20
define e(x) {
    auto a, b, c, i, s
    a = 1
    b = 1
    s = 1
    for(i=1; l==1; i++) {
        a = a*x
        b = b*i
        c = a/b
        if(c == 0) return(s)
        s = s+c
    }
}
```

EXAMPLE 3 Printing the approximate values of the function

Prints approximate values of the exponential function of the first ten integers:

```bash
for(i=1; i<=10; i++) e(i)
```

or

```bash
for (i = 1; i <= 10; ++i) { e(i) }
```

ENVIRONMENT VARIABLES
See environ(5) for descriptions of the following environment variables that affect the execution of bc: LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS
The following exit values are returned:

- **0**: All input files were processed successfully.
- **unspecified**: An error occurred.

FILES
- `/usr/lib/lib.b`: mathematical library
- `/usr/include/limits.h`: to define BC_ parameters

ATTRIBUTES
See attributes(5) for descriptions of the following attributes:
The `bc` command does not recognize the logical operators `&&` and `||`.

The `for` statement must have all three expressions (`E's`).

**SEE ALSO**
dc(1), awk(1), attributes(5)

**NOTES**
The `bc` command does not recognize the logical operators `&&` and `||`.

The `for` statement must have all three expressions (`E's`).
bdiff

NAME
bdiff – big diff

SYNOPSIS
bdiff filename1 filename2 [n] [-s]

DESCRIPTION
bdiff is used in a manner analogous to diff to find which lines in filename1 and
filename2 must be changed to bring the files into agreement. Its purpose is to allow
processing of files too large for diff. If filename1 (filename2) is –, the standard input is
read.

bdiff ignores lines common to the beginning of both files, splits the remainder of
each file into n-line segments, and invokes diff on corresponding segments. If both
optional arguments are specified, they must appear in the order indicated above.

The output of bdiff is exactly that of diff, with line numbers adjusted to account
for the segmenting of the files (that is, to make it look as if the files had been processed
whole). Note: Because of the segmenting of the files, bdiff does not necessarily find a
smallest sufficient set of file differences.

OPTIONS
n The number of line segments. The value of n is 3500 by default. If the
optional third argument is given and it is numeric, it is used as the value
for n. This is useful in those cases in which 3500-line segments are too large
for diff, causing it to fail.

-s Specifies that no diagnostics are to be printed by bdiff (silent option).
Note: However, this does not suppress possible diagnostic messages from
diff, which bdiff calls.

USAGE
See largefile(5) for the description of the behavior of bdiff when encountering
files greater than or equal to 2 Gbyte (2^{31} bytes).

FILES
/tmp/bd?????

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

SEE ALSO
diff(1), attributes(5), largefile(5)

DIAGNOSTICS
Use help for explanations.
The \texttt{bfs} command is (almost) like \texttt{ed(1)} except that it is read-only and processes much larger files. Files can be up to 1024K bytes and 32K lines, with up to 512 characters, including new-line, per line (255 for 16-bit machines). \texttt{bfs} is usually more efficient than \texttt{ed(1)} for scanning a file, since the file is not copied to a buffer. It is most useful for identifying sections of a large file where \texttt{csplit(1)} can be used to divide it into more manageable pieces for editing.

Normally, the size of the file being scanned is printed, as is the size of any file written with the \texttt{w (write)} command. The optional – suppresses printing of sizes. Input is prompted with * if P and a carriage return are typed, as in \texttt{ed(1)}. Prompting can be turned off again by inputting another P and carriage return. Note that messages are given in response to errors if prompting is turned on.

All address expressions described under \texttt{ed(1)} are supported. In addition, regular expressions may be surrounded with two symbols besides / and ?:

\begin{itemize}
  \item > indicates downward search without wrap-around, and
  \item < indicates upward search without wrap-around.
\end{itemize}

There is a slight difference in mark names; that is, only the letters a through z may be used, and all 26 marks are remembered.

\textbf{bfs Commands}

The e, g, v, k, p, q, w, =, !, and null commands operate as described under \texttt{ed(1)}. Commands such as --, +++-, ++++, -12, and +4p are accepted. Note that 1,10p and 1,10 will both print the first ten lines. The f command only prints the name of the file being scanned; there is no remembered file name. The w command is independent of output diversion, truncation, or crunching (see the xo, xt, and xc commands, below). The following additional commands are available:

\begin{itemize}
  \item \texttt{x f file}
    Further commands are taken from the named \texttt{file}. When an end-of-file is reached, an interrupt signal is received or an error occurs, reading resumes with the file containing the xf. The xf commands may be nested to a depth of 10.
  \item \texttt{x n}
    List the marks currently in use (marks are set by the k command).
  \item \texttt{x o [ file ]}
    Further output from the p and null commands is diverted to the named \texttt{file}, which, if necessary, is created mode 666 (readable and writable by everyone), unless your \texttt{umask} setting (see \texttt{umask(1)}) dictates otherwise. If \texttt{file} is missing, output is diverted to the standard output. Note that each diversion causes truncation or creation of the file.
\end{itemize}
: label
This positions a label in a command file. The label is terminated by new-line, and
blanks between the : (colon) and the start of the label are ignored. This command
may also be used to insert comments into a command file, since labels need not be
referenced.

(. , . )xb/regular expression/label
A jump (either upward or downward) is made to label if the command succeeds. It
fails under any of the following conditions:

1. Either address is not between 1 and $.
2. The second address is less than the first.
3. The regular expression does not match at least one line in the specified range,
   including the first and last lines.

On success, . (dot) is set to the line matched and a jump is made to label. This
command is the only one that does not issue an error message on bad addresses, so
it may be used to test whether addresses are bad before other commands are
executed. Note that the command, xb/^/ label1, is an unconditional jump.

The xb command is allowed only if it is read from someplace other than a terminal.
If it is read from a pipe, only a downward jump is possible.

xt: number
Output from the p and null commands is truncated to, at most, number characters.
The initial number is 255.

xv[digit] [spaces] [value]
The variable name is the specified digit following the xv. The commands xv5100
or xv5 100 both assign the value 100 to the variable 5. The command xv61,100p
assigns the value 1,100p to the variable 6. To reference a variable, put a % in front
of the variable name. For example, using the above assignments for variables 5 and
6:

1,%5p
1,%5
%6

will all print the first 100 lines.

g/%5/p

would globally search for the characters 100 and print each line containing a
match. To escape the special meaning of %, a \ must precede it.

g/\.*\%[cds]/p
could be used to match and list %c, %d, or %s formats (for example, "printf"-like statements) of characters, decimal integers, or strings. Another feature of the xv command is that the first line of output from a UNIX system command can be stored into a variable. The only requirement is that the first character of value be an !. For example:

```
.w junk
xv5!cat junk
!rm junk
!echo "%5"
```

would put the current line into variable 35, print it, and increment the variable 36 by one. To escape the special meaning of ! as the first character of value, precede it with a \\.

```
xv7\!date
```

stores the value !date into variable 7.

```
xbz label
xbn label
```

These two commands will test the last saved return code from the execution of a UNIX system command (!command) or nonzero value, respectively, to the specified label. The two examples below both search for the next five lines containing the string size:

Example 1:

```
xv5
: l
/size/
xv5!expr %5 - 1
!if 0%5 != 0 exit 2
xbn 1
```

Example 2:

```
xv4
: l
/size/
xv4!expr %4 - 1
!if 0%4 = 0 exit 2
xbz 1
```

```
xc [switch]
```

If switch is 1, output from the p and null commands is crunched; if switch is 0, it is not. Without an argument, xc reverses switch. Initially, switch is set for no crunching. Crunched output has strings of tabs and blanks reduced to one blank and blank lines suppressed.

**OPERANDS**

The following operand is supported:
filename

Any file up to 1024K bytes and 32K lines, with up to 512 characters, including new-line, per line (255 for 16-bit machines). filename can be a section of a larger file which has been divided into more manageable sections for editing by the use of csplit(1).

EXIT STATUS

The following exit values are returned:

0 Successful completion without any file or command errors.

>0 An error occurred.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
</tbody>
</table>

SEE ALSO

csplit(1), ed(1), umask(1), attributes(5)

DIAGNOSTICS

Message is ? for errors in commands, if prompting is turned off. Self-explanatory error messages are displayed when prompting is on.
NAME  biff – give notice of incoming mail messages

SYNOPSIS  /usr/ucb/biff [y | n]

DESCRIPTION  biff turns mail notification on or off for the terminal session. With no arguments, biff displays the current notification status for the terminal.

If notification is allowed, the terminal rings the bell and displays the header and the first few lines of each arriving mail message. biff operates asynchronously. For synchronized notices, use the MAIL variable of sh(1) or the mail variable of csh(1).

A ‘biff y’ command can be included in your ~/.login or ~/.profile file for execution when you log in.

OPTIONS  
  y  Allow mail notification for the terminal.
  n  Disable notification for the terminal.

FILES  
  ~/.login  User’s login file
  ~/.profile  User’s profile file

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

SEE ALSO  csh(1), mail(1), sh(1), attributes(5)
NAME

break, continue – shell built-in functions to escape from or advance within a controlling while, for, foreach, or until loop

sh

break [n]
continue [n]

csh

break
continue

ksh

*break [n]
*continue [n]

sh

break exits from the enclosing for or while loop, if any. If n is specified, break n levels.

continue resumes the next iteration of the enclosing for or while loop. If n is specified, resume at the n-th enclosing loop.

csh

break resumes execution after the end of the nearest enclosing foreach or while loop. The remaining commands on the current line are executed. This allows multilevel breaks to be written as a list of break commands, all on one line.

continue continues execution of the next iteration of the nearest enclosing while or foreach loop.

ksh

break exits from the enclosed for, while, until, or select loop, if any. If n is specified then break n levels.

continue resumes the next iteration of the enclosed for, while, until, or select loop. If n is specified then resume at the n-th enclosed loop.

On this man page, ksh(1) commands that are preceded by one or two * (asterisks) are treated specially in the following ways:

1. Variable assignment lists preceding the command remain in effect when the command completes.
2. I/O redirections are processed after variable assignments.
3. Errors cause a script that contains them to abort.
4. Words, following a command preceded by ** that are in the format of a variable assignment, are expanded with the same rules as a variable assignment. This means that tilde substitution is performed after the = sign and word splitting and file name generation are not performed.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:
See Also: csh(1), exit(1), ksh(1), sh(1), attributes(5)
The `cal` utility writes a Gregorian calendar to standard output. If the `year` operand is specified, a calendar for that year is written. If no operands are specified, a calendar for the current month is written.

The following operands are supported:

| `month` | Specify the month to be displayed, represented as a decimal integer from 1 (January) to 12 (December). The default is the current month. |
| `year` | Specify the year for which the calendar is displayed, represented as a decimal integer from 1 to 9999. The default is the current year. |

See `environ(5)` for descriptions of the following environment variables that affect the execution of `cal`: `LC_TIME`, `LC_MESSAGES`, and `NLSPATH`.

The following exit values are returned:

- 0: Successful completion.
- >0: An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
</tbody>
</table>

An unusual calendar is printed for September 1752. That is the month 11 days were skipped to make up for lack of leap year adjustments. To see this calendar, type:

```
cal 9 1752
```

The command `cal 83` refers to the year 83, not 1983.

The year is always considered to start in January.

SEE ALSO

`calendar(1)`, `attributes(5)`, `environ(5)`
calendar utility consults the file `calendar` in the current directory and writes lines that contain today’s or tomorrow’s date anywhere in the line to standard output. Most reasonable month-day dates such as Aug. 24, august 24, 8/24, and so forth, are recognized, but not 24 August or 24/8. On Fridays and weekends “tomorrow” extends through Monday. `calendar` can be invoked regularly by using the `crontab(1)` or `at(1)` commands.

When the optional argument `-` is present, `calendar` does its job for every user who has a file `calendar` in his or her login directory and sends them any positive results by `mail(1)`. Normally this is done daily by facilities in the UNIX operating system (see `cron(1M)`).

If the environment variable `DATEMSK` is set, `calendar` will use its value as the full path name of a template file containing format strings. The strings consist of conversion specifications and text characters and are used to provide a richer set of allowable date formats in different languages by appropriate settings of the environment variable `LANG` or `LC_TIME`; see `environ(5)`. See `strftime(3C)` for the list of allowable conversion specifications.

**EXAMPLE 1** Possible contents of a template

The following example shows the possible contents of a template:

```bash
%B %eth of the year %Y
```

 `%B` represents the full month name, %e the day of month and %Y the year (4 digits).

If `DATEMSK` is set to this template, the following `calendar` file would be valid:

```bash
March 7th of the year 1989 <Reminder>
```

See `environ(5)` for descriptions of the following environment variables that affect the execution of `calendar`: `LC_CTYPE`, `LC_TIME`, `LC_MESSAGES`, `NLSPATH`, and `TZ`.

**FILES**

- `/etc/passwd` system password file
- `/tmp/cal*` temporary files used by `calendar`
- `/usr/lib/calprog` program used to determine dates for today and tomorrow

**EXIT STATUS**

- 0 Successful completion.
- >0 An error occurred.

**ATTRIBUTES**

See `attributes(5)` for descriptions of the following attributes:
calendar(1)

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
</tbody>
</table>

SEE ALSO at(1), crontab(1), mail(1), cron(1M), ypbind(1M), strftime(3C), attributes(5), environ(5)

NOTES Appropriate lines beginning with white space will not be printed.

Your calendar must be public information for you to get reminder service.

calendar’s extended idea of “tomorrow” does not account for holidays.

The - argument works only on calendar files that are local to the machine; calendar is intended not to work on calendar files that are mounted remotely with NFS. Thus, `calendar -` should be run only on diskful machines where home directories exist; running it on a diskless client has no effect.

calendar is no longer in the default root crontab. Because of the network burden `calendar -` can induce, it is inadvisable in an environment running ypbind(1M) with a large passwd.byname map. If, however, the usefulness of calendar outweighs the network impact, the super-user may run `crontab -e` to edit the root crontab. Otherwise, individual users may wish to use `crontab -e` to edit their own crontabs to have cron invoke calendar without the - argument, piping output to mail addressed to themselves.
cancel(1)

NAME
cancel – cancel print request

SYNOPSIS
cancel [request-ID...] [destination...]
cancel -u user... [destination...]

DESCRIPTION
The cancel utility cancels print requests. There are two forms of the cancel
command.

The first form of cancel has two optional arguments: print requests (request-ID) and
destinations (destination). Specifying request-ID with destination cancels request-ID on
destination. Specifying only the destination cancels the current print request on
destination. If destination is not specified, cancel cancels the requested print request on
all destinations.

The second form of cancel cancels a user’s print requests on specific destinations.

Users can only cancel print requests associated with their username. By default, users
can only cancel print requests on the host from which the print request was submitted.
If a super-user has set user-equivalence=true in /etc/printers.conf on the
print server, users can cancel print requests associated with their username on any
host. Super-users can cancel print requests on the host from which the print request
was submitted. Super-users can also cancel print requests from the print server.

The print client commands locate destination information using the “printers”
database in the name service switch. See nsswitch.conf(4), printers(4), and
printers.conf(4) for details.

OPTIONS
The following options are supported:

- u user
  The name of the user for which print requests are to be cancelled.
  Specify user as a username.

OPERANDS
The following operands are supported:

destination
  The destination on which the print requests are to be canceled.
  destination is the name of a printer or class of printers (see
  ladmin(1M)). If destination is not specified, cancel cancels the
  requested print request on all destinations. Specify destination
  using atomic, POSIX-style (server:destination), or Federated
  Naming Service (FNS) (. . ./service/printer/ . . )
  names. See NOTES for information regarding using POSIX-style
  destination names with cancel. See printers.conf(4) for
  information regarding the naming conventions for atomic and FNS
  names, and standards(5) for information regarding POSIX.

request-ID
  The print request to be canceled. Specify request-ID using LP-style
  request IDs (destination-number).

user
  The name of the user for which the print requests are to be
cancelled. Specify user as a username.

User Commands 109
The following exit values are returned:

0 Successful completion.
non-zero An error occurred.

FILES
/var/spool/print/* LP print queue.
$HOME/.printers User-configurable printer database.
/etc/printers.conf System printer configuration database.
printers.conf.byname NIS version of /etc/printers.conf.
printers.org_dir NIS+ version of /etc/printers.conf.
fns.ctx_dir.domain FNS version of /etc/printers.conf.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWpcu</td>
</tr>
</tbody>
</table>

SEE ALSO
lp(1), lpg(1B), lpr(1B), lprm(1B), lpstat(1), lpadmin(1M), nsswitch.conf(4), printers(4), printers.conf(4), attributes(5), standards(5)

NOTES
POSIX-style destination names (server:destination) are treated as print requests if destination has the same format as an LP-style request-ID. See standards(5).
NAME
cat – concatenate and display files

SYNOPSIS
cat [-nbsuvet] [file...]

DESCRIPTION
The cat utility reads each file in sequence and writes it on the standard output. Thus:
example% cat file
prints file on your terminal, and:
example% cat file1 file2 >file3
concatenates file1 and file2, and writes the results in file3. If no input file is given, cat reads from the standard input file.

OPTIONS
The following options are supported:

- n
  Precede each line output with its line number.
- b
  Number the lines, as - n, but omit the line numbers from blank lines.
- u
  The output is not buffered. (The default is buffered output.)
- s
  cat is silent about non-existent files.
- v
  Non-printing characters (with the exception of tabs, new-lines and form-feeds) are printed visibly. ASCII control characters (octal 000 – 037) are printed as ^n, where n is the corresponding ASCII character in the range octal 100 – 137 (0, A, B, C, . . ., X, Y, Z, [, \, ], ^, and _); the DEL character (octal 0177) is printed ^?. Other non-printable characters are printed as M-x, where x is the ASCII character specified by the low-order seven bits.

When used with the -v option, the following options may be used:

- e
  A $ character will be printed at the end of each line (prior to the new-line).
- t
  Tabs will be printed as ^I’s and formfeeds to be printed as ^L’s.

The -e and -t options are ignored if the -v option is not specified.

OPERANDS
The following operand is supported:

file
A path name of an input file. If no file is specified, the standard input is used. If file is ‘ ’, cat will read from the standard input at that point in the sequence. cat will not close and reopen standard input when it is referenced in this way, but will accept multiple occurrences of ‘ ’ as file.

EXAMPLES
EXAMPLE 1 Concatenating a file
The following command:
example% cat myfile
EXAMPLE 1 Concatenating a file  (Continued)

writes the contents of the file myfile to standard output.

EXAMPLE 2 Concatenating two files into one

The following command:

```
example% cat doc1 doc2 > doc.all
```

concatenates the files doc1 and doc2 and writes the result to doc.all.

EXAMPLE 3 Concatenating two arbitrary pieces of input with a single invocation

The command:

```
example% cat start − middle − end > file
```

when standard input is a terminal, gets two arbitrary pieces of input from the terminal with a single invocation of cat. Note, however, that if standard input is a regular file, this would be equivalent to the command:

```
cat start − middle /dev/null end > file
```

because the entire contents of the file would be consumed by cat the first time ‘−’ was used as a file operand and an end-of-file condition would be detected immediately when ‘−’ was referenced the second time.

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of cat: LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS

The following exit values are returned:

- 0  All input files were output successfully.
- >0  An error occurred.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

SEE ALSO

touch(1), environ(5), attributes(5), largefile(5)

NOTES

Redirecting the output of cat onto one of the files being read will cause the loss of the data originally in the file being read. For example,

```
example% cat filename1 filename2 >filename1
```

causes the original data in filename1 to be lost.
## NAME
cc - C compiler

## SYNOPSIS
/usr/ucb/cc [options]

## DESCRIPTION
/usr/ucb/cc is the interface to the BSD Compatibility Package C compiler. It is a
script that looks for the link /usr/ccs/bin/ucbcc to the C compiler. The
/usr/ccs/bin/ucbcc link is available only with the SPROcc package, whose
default location is /opt/SUNWspro. The /usr/ucb/cc interface is identical to
/usr/ccs/bin/ucbcc, except that BSD headers are used and BSD libraries are
linked before base libraries. The /opt/SUNWspro/man/man1/acc.1 man page is
available only with the SPROcc package.

## OPTIONS
The /usr/ucb/cc interface accepts the same options as /usr/ccs/bin/ucbcc,
with the following exceptions:

- **-I dir**
  Search dir for included files whose names do not begin with a
  slash (/) prior to searching the usual directories. The directories
  for multiple -I options are searched in the order specified. The
  preprocessor first searches for include files in the directory
  containing sourcefile, and then in directories named with -I
  options (if any), then /usr/ucbinclude, and finally, in
  /usr/include.

- **-L dir**
  Add dir to the list of directories searched for libraries by
  /usr/ccs/bin/ucbcc. This option is passed to
  /usr/ccs/bin/ld and /usr/lib. Directories specified with this
  option are searched before /usr/ucblib and /usr/lib.

- **-Y P, dir**
  Change the default directory used for finding libraries.

## EXIT STATUS
The following exit values are returned:

0  Successful compilation or link edit.

>0  An error occurred.

## FILES
/usr/ccs/bin/ld link editor
/usr/lib/libc C library
/usr/ucbinclude BSD Compatibility directory for header files
/usr/ucblib BSD Compatibility directory for libraries
/usr/ucblib/libucb BSD Compatibility C library
/usr/lib/libsocket library containing socket routines
/usr/lib/libnsl library containing network functions
/usr/lib/libelf library containing routines to process ELF object files
/usr/lib/libaio library containing asynchronous I/O routines

## ATTRIBUTES
See attributes(5) for descriptions of the following attributes:
cc(1B)

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

**SEE ALSO**  
ld(1), a.out(4), attributes(5)

**NOTES**  
The `-y P, dir` option may have unexpected results and should not be used.
cd(1)

NAME      cd, chdir, pushd, popd, dirs - change working directory
SYNOPSIS /

/sh 

In contrast to the version built into the shell, as described below. /

/cdrn/cd has no effect on the invoking process but can be used to determine whether or not a given directory can be set as the current directory.

/sh The Bourne shell built-in cd changes the current directory to argument. The shell parameter HOME is the default argument. The shell parameter CDPATH defines the search path for the directory containing argument. Alternative directory names are separated by a colon (:\). The default path is <null> (specifying the current directory). Note: The current directory is specified by a null path name, which can appear immediately after the equal sign or between the colon delimiters anywhere else in the path list. If argument begins with ‘/’, ‘.’, or ‘..’, the search path is not used. Otherwise, each directory in the path is searched for argument. cd must have execute (search) permission in argument. Because a new process is created to execute each command, cd would be ineffective if it were written as a normal command; therefore, it is recognized by and is internal to the shell. (See pwd(1), sh(1), and chdir(2)).

chdir is just another way to call cd.

csh If dir is not specified, the C shell built-in cd uses the value of shell parameter HOME as the new working directory. If dir specifies a complete path starting with ‘/’, ‘.’, or ‘..’, dir becomes the new working directory. If neither case applies, cd tries to find the designated directory relative to one of the paths specified by the CDPATH shell variable. CDPATH has the same syntax as, and similar semantics to, the PATH shell variable. cd must have execute (search) permission in dir. Because a new process is created to execute each command, cd would be ineffective if it were written as a normal command; therefore, it is recognized by and is internal to the C-shell. (See pwd(1), sh(1), and chdir(2)).

User Commands 115
chdir changes the shell’s working directory to directory dir. If no argument is given, change to the home directory of the user. If dir is a relative pathname not found in the current directory, check for it in those directories listed in the cdpath variable. If dir is the name of a shell variable whose value starts with a /, change to the directory named by that value.

pushd will push a directory onto the directory stack. With no arguments, exchange the top two elements.

+n  Rotate the n’th entry to the top of the stack and cd to it.
dir  Push the current working directory onto the stack and change to dir.

popd pops the directory stack and cd to the new top directory. The elements of the directory stack are numbered from 0 starting at the top.

+n  Discard the n’th entry in the stack.

dirs will print the directory stack, most recent to the left; the first directory shown is the current directory. With the -l argument, produce an unabbreviated printout; use of the - notation is suppressed.

ksh

The Korn shell built-in cd command can be in either of two forms. In the first form it changes the current directory to arg. If arg is – the directory is changed to the previous directory. The shell variable HOME is the default arg. The variable PWD is set to the current directory. The shell variable CDPATH defines the search path for the directory containing arg. Alternative directory names are separated by a colon (:). The default path is <null> (specifying the current directory). Note that the current directory is specified by a null path name, which can appear immediately after the equal sign or between the colon delimiters anywhere else in the path list. If arg begins with a ‘/’, ‘.’, ‘.’’, or ‘. . .’, then the search path is not used. Otherwise, each directory in the path is searched for arg.

The second form of cd substitutes the string new for the string old in the current directory name, PWD and tries to change to this new directory.

The cd command may not be executed by zsh. Because a new process is created to execute each command, cd would be ineffective if it were written as a normal command; therefore, it is recognized by and is internal to the Korn shell. (See pwd(1), sh(1), and chdir(2)).

OPERANDS

The following operands are supported:

directory  An absolute or relative pathname of the directory that becomes the new working directory. The interpretation of a relative pathname by cd depends on the CDENV environment variable.

OUTPUT

If a non-empty directory name from CDENV is used, an absolute pathname of the new working directory will be written to the standard output as follows:

"$s\n", <new directory>
Otherwise, there will be no output.

**ENVIRONMENT VARIABLES**

See `environ(5)` for descriptions of the following environment variables that affect the execution of `cd`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

**CDPATH**
A colon-separated list of pathnames that refer to directories. If the directory operand does not begin with a slash ( `/` ) character, and the first component is not dot or dot-dot, `cd` will search for directory relative to each directory named in the `CDPATH` variable, in the order listed. The new working directory will be set to the first matching directory found. An empty string in place of a directory pathname represents the current directory. If `CDPATH` is not set, it will be treated as if it were an empty string.

**HOME**
The name of the home directory, used when no directory operand is specified.

**PWD**
A pathname of the current working directory, set by `cd` after it has changed to that directory.

**EXIT STATUS**
The following exit values are returned by `cd`:

<table>
<thead>
<tr>
<th>Exit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The directory was successfully changed.</td>
</tr>
<tr>
<td>&gt;0</td>
<td>An error occurred.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Attribute Type</th>
<th>Attribute Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

**SEE ALSO**
`csh(1)`, `ksh(1)`, `pwd(1)`, `sh(1)`, `chdir(2)`, `attributes(5)`, `environ(5)`
The `cdrw` command provides the ability to create data and audio CDs. It also provides the ability to extract audio tracks from an audio CD. Any MMC-compliant CD-R or CD-RW drive can be used with `cdrw`.

`cdrw` will search for a CD writer device connected to the system, unless the user specifies a device with the `-d` option. If it finds a single such writer device, it will use that as the default CD writer device for the command.

When more than one CD writer is connected to the system, use the `-d` option to indicate which device is desired. The device name can be specified in one of the following ways: `/dev/rdsk/cNtNdNsN`, `cNtNdNsN`, `cNtNdN`, or a symbolic name used by volume manager, such as `cdrom` or `cdrom1`. The `-l` option will provide a list of CD writers.

For instructions on adding a USB-mass-storage-class-compliant CD-RW to your system, see `scs2usb(7D)`.

### Creating Data CDs

When creating data CDs, `cdrw` uses the track-at-once mode of writing. With the `-i` option, the user will specify a file that contains the data to write on CD media. In the absence of such a file, `cdrw` will read data from standard input.

In either case, the data will typically first have been prepared by using the `mkisofs(1M)` command to convert the file and file information into the High Sierra format used on CDs. See the examples that include use of this command.

### Creating Audio CDs

For creating an audio CD, using the `-a` option, single or multiple audio files can be specified. All of the audio files should be in the supported audio formats. Currently approved formats are:

- **sun** `Sun .au` files with data in Red Book CDDA form
- **wav** RIFF (.wav) files with data in Red Book CDDA format
- **cda** `.cda` files having raw CD audio data (that is, 16 bit PCM stereo at 44.1 KHz sample rate in little-endian byteorder)
aur .aur files having raw CD data in big-endian byteorder

If no audio format is specified, cdrw tries to understand the audio file format based on the file extension. The case of the characters in the extension is ignored. If a format is specified using the -T option, it will be assumed as the audio file type for all the files specified. Also, -cdrw will close the session after writing the audio tracks. Therefore, the tracks to be written should be specified in a single command line.

Extracting Audio
cdrw can also be used for extracting audio data from an audio CD with the -x option. The CD should have tracks in Red Book CDDA form. By default, the output format is based on the file extension. A user can specify a sun, wav, cda, or aur output format using the -T option.

Copying CDs
cdrw can be used to copy single session data CD-ROMs and Red Book audio CDs. For copying a CD, cdrw looks for a specified source device. If no source device is specified when using the -c option, the current CD writing device is assumed to be the source. cdrw will extract the track or tracks into a temporary file and will look for a blank writable CD-R/RW media in the current CD writing device. If no such media is found, the user will be asked to insert a blank writable CD media in the current CD writing device. If enough space is not available in the default temporary directory, an alternative directory can be specified using the -m option.

Erasing CD-RW Media
Users have to erase the CD-RW media before it can be re-written. With the -b option, the following flavors of erasing are currently supported:

```
session   Erase the last session.
all       Erase the entire media.
```

If the session erasing type is used, cdrw will erase the last session. If there is only one session recorded on the CD-RW (for example, a data/audio CD-RW created by this tool), then session erasing is useful as it will only erase the portion that is recorded, leaving behind a blank disk. This is faster than erasing the entire media.

The all erasing type should be used if it is a multisession disk, or the last session is not closed, or disk status is unknown, and the user wishes to erase the disk. With this type of erase, cdrw will erase the entire disk.

Checking device-list or media-status
The user can get a list of CD writing devices currently present in the system with the -l option. Also, for a particular media, the user can get the blanking status and table of contents through the -M option. The -M option also prints information about the last session start address and the next writable address. This information, along with the -o option, can be used to create multisession CDs. Please refer to mkisofs(1M) for more information.

OPTIONS
The following options are supported:

```
-a     Creates an audio disk. At least one audio-file name must be specified. A CD can not have more than 99 audio tracks, so no more than 99 audio files can
```
be specified. Also, the maximum audio data that can be written to the
media by default is 74 minutes, unless -c is specified.
-b Blanks a CD-RW media. The type of erasing must be specified by the all
or session argument.
-c Copies a CD. If no other argument is specified, the default CD writing
device is assumed to be the source device as well. In this case, the copying
operation will read the source media into a temporary directory and will
prompt the user to place a blank media into the drive for copying to
proceed.
-C Uses media stated capacity. Without this option, cdrw will use a default
value for writable CD media, which is 74 minutes for an audio CD or
681984000 bytes for a data CD.
-d Specifies CD writing device.
-h Help. Prints usage message.
-i Specifies image file for creating data CDs. The file size should be less than
what can be written on a CD-R or CD-RW media, which is 681984000 bytes
by default or the media stated capacity if the -C option is used. Also, it is
better to have the file locally available instead of having it on an
NFS-mounted filesystem, because the CD writing process expects data to
be available continuously without interruptions.
-l Lists all the CD writers found in the system.
-m Uses an alternate temporary directory instead of system default temporary
directory for storing track data while copying a CD. An alternate
temporary directory might be required because the amount of data on a CD
can be huge (as much as 800 Mbytes for an 80 minute audio CD) and the
system default temporary directory might not have that much space.
-M Reports media status. cdrw will report if the media is blank or not, its table
of contents, the last session’s start address, and the next writable address if
the disk is open.
-O Keeps the disk open. cdrw will close the session, but it will keep the disk
open so that another session can be added later on to create a multisession
disk.
-p Sets the CD writing speed. For example, -p 4 will set the speed to 4X. If
this option is not specified, cdrw will use the default speed of the CD
writer. If this option is specified, cdrw will try to set the drive write speed
to this value, but there is no guarantee of the speed actually used by the
drive.
-s Specifies source device for copying CD.
Simulation mode. In this mode, `cdrw` will do everything with the drive laser turned off, so nothing will be written to the media. This can be used to verify if the system can provide data at a rate good enough for CD writing.

Audio format to use extracting audio files or reading audio files for audio CD creation. The `audio-type` can be `sun`, `wav`, `cda`, or `aur`.

Verbose mode.

Extracts audio data from an audio track.

**EXAMPLE 1** Creating a data CD
```
example% cdrw -i /local/iso_image
```

**EXAMPLE 2** Creating a CD from a directory
This example creates a CD from the directory tree `/home/foo`:
```
example% mkisofs -r /home/foo 2>/dev/null | cdrw -i -p1
```

**EXAMPLE 3** Extracting an audio track number
This example extracts audio track number 1 to `/home/foo/song1.wav`:
```
example% cdrw -x -T wav 1 /home/foo/song1.wav
```

**EXAMPLE 4** Using `wav` files
This example creates an audio CD from `wav` files on disk:
```
example% cdrw -a song1.wav song2.wav song3.wav song4.wav
```

**EXAMPLE 5** Erasing a CD-RW media
This example erases a CD-RW media in a CD-RW drive:
```
example% cdrw -b all
```

**EXAMPLE 6** Creating a data CD with multiple drives
This example creates a data CD on a system with multiple CD-R/RW drives:
```
example% cdrw -d clt6d0s2 -i /home/foo/iso-image
```

**EXAMPLE 7** Checking data delivery rate
This example checks if the system can provide data to a CD-RW drive at a rate sufficient for the write operation:
EXAMPLE 7 Checking data delivery rate (Continued)

eample$ cdrw -S -i /home/foo/iso-image

EXAMPLE 8 Running at a higher priority

This example runs cdrw at a higher priority (for root user only):

eample# priocntl -e -p 60 cdrw -i /home/foo/iso-image

EXAMPLE 9 Creating a multi-session disk

Create the first session image using mkisofs(1M) and record it onto the disk without closing the disk:

eample$ cdrw -O -i /home/foo/iso-image

Additional sessions can be added to an open disk by creating an image with mkisofs(1M) using the session start and next writable address reported by cdrw.

eample$ cdrw -M

<table>
<thead>
<tr>
<th>Track No.</th>
<th>Type</th>
<th>Start address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Data</td>
<td>0</td>
</tr>
<tr>
<td>Leadout</td>
<td>Data</td>
<td>166564</td>
</tr>
</tbody>
</table>

Last session start address: 162140
Next writable address: 173464

example$ mkisofs -o /tmp/image2 -r -C 0,173464 -M \
/dev/rdsk/c0t2d0s2 /home/foo

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcdrw</td>
</tr>
</tbody>
</table>

SEE ALSO

audioconvert(1), mkisofs(1M), priocntl(1), attributes(5), rbac(5), scsa2usb(7D), sd(7D)

NOTES

The CD writing process requires data to be supplied at a constant rate to the drive. It is advised to keep I/O activity to a minimum and shut down the related applications while writing CDs.

When making copies or extracting audio tracks, it is better to use an MMC compliant source CD-ROM drive. The CD writing device can be used for this purpose.
Before writing a CD, ensure that the media is blank by using the -M option and use the -S simulation mode to test the system to make sure it can provide data at the required rate. In case the system is not able to provide data at the required rate, try simulation with a slower write speed set through the -p option. Users can also try to run cdrw at a higher priority using the priocntl(1) command.

The -p option is provided for users who are aware of the CD-R/RW drive and its capabilities to operate at different write speeds. Some commercially available drives handle the drive speed setting command differently, so use this option judiciously.

Most commercially available drives allow writing beyond 74 minutes as long as the media has the capacity (such as 80-minute media). However, such capability of writing beyond 74 minutes might not be supported by the drive in use. If the drive being used supports such capability, then use the -C option to indicate that the tool should rely on the capacity indicated by the media.

The cdrw command uses rbac(5) to control user access to the devices. By default, cdrw is accessible to all users but can be restricted to individual users. Please refer to "Administering CD-R/CD-RW devices" in the System Administration Guide: Basic Administration for more information.
checknr(1)

NAME    checknr – check nroff and troff input files; report possible errors

SYNOPSIS

checknr [-fs] [-a .x1 .y1 .x2 .y2 ... .xn .yn] [-c .x1 .x2 .x3 ... .xn] [filename...]

DESCRIPTION

checknr checks a list of nroff(1) or troff(1) input files for certain kinds of errors
involving mismatched opening and closing delimiters and unknown commands. If no
files are specified, checknr checks the standard input. Delimiters checked are:

- Font changes using \fx ... \fP.
- Size changes using \sx ... \s0.
- Macros that come in open ... close forms, for example, the .TS and .TE macros
  which must always come in pairs.

checknr knows about the ms(5) and me(5) macro packages.

checknr is intended to be used on documents that are prepared with checknr in
mind. It expects a certain document writing style for \f and \s commands, in that
each \fx must be terminated with \fP and each \sx must be terminated with \s0.
While it will work to directly go into the next font or explicitly specify the original font
or point size, and many existing documents actually do this, such a practice will
produce complaints from checknr. Since it is probably better to use the \fP and \s0
forms anyway, you should think of this as a contribution to your document
preparation style.

OPTIONS

- f     Ignore \f font changes.
- s     Ignore \s size changes.
- a .x1 .y1 ... Add pairs of macros to the list. The pairs of macros are assumed to
  be those (such as .DS and .DE) that should be checked for
  balance. The -a option must be followed by groups of six
  characters, each group defining a pair of macros. The six characters
  are a period, the first macro name, another period, and the second
  macro name. For example, to define a pair .BS and .ES, use
  '-a .BS .ES'
- c .x1 ... Define commands which checknr would otherwise complain
  about as undefined.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWdoc</td>
</tr>
</tbody>
</table>

SEE ALSO    eqn(1), nroff(1), troff(1), attributes(5), me(5), ms(5)

BUGS        There is no way to define a one-character macro name using the -a option.
chgrp – change file group ownership

**SYNOPSIS**

```
chgrp [-fhR] group file...
```

**DESCRIPTION**

The `chgrp` utility will set the group ID of the file named by each `file` operand to the group ID specified by the `group` operand.

For each `file` operand, it will perform actions equivalent to the `chown(2)` function, called with the following arguments:

- The `file` operand will be used as the `path` argument.
- The user ID of the file will be used as the `owner` argument.
- The specified group ID will be used as the `group` argument.

Unless `chgrp` is invoked by a process with appropriate privileges, the set-user-ID and set-group-ID bits of a regular file will be cleared upon successful completion; the set-user-ID and set-group-ID bits of other file types may be cleared.

The operating system has a configuration option `_POSIX_CHOWN_RESTRICTED`, to restrict ownership changes. When this option is in effect, the owner of the file may change the group of the file only to a group to which the owner belongs. Only the super-user can arbitrarily change owner IDs, whether or not this option is in effect. To set this configuration option, include the following line in `/etc/system`:

```
set rstchown = 1
```

To disable this option, include the following line in `/etc/system`:

```
set rstchown = 0
```

_`_POSIX_CHOWN_RESTRICTED_`_ is enabled by default. See `system(4)` and `fpathconf(2)`.

**OPTIONS**

- `-f` Force. Do not report errors.
- `-h` If the file is a symbolic link, change the group of the symbolic link. Without this option, the group of the file referenced by the symbolic link is changed.
- `-R` Recursive. `chgrp` descends through the directory, and any subdirectories, setting the specified group ID as it proceeds. When a symbolic link is encountered, the group of the target file is changed (unless the `-h` option is specified), but no recursion takes place.

**OPERANDS**

The following operands are supported:

- `group` A group name from the group database or a numeric group ID. Either specifies a group ID to be given to each file named by one of the `file` operands. If a numeric `group` operand exists in the group database as a group name, the group ID number associated with that group name is used as the group ID.

- `file` A path name of a file whose group ID is to be modified.
chgrp(1)

**Usage**
See largefile(5) for the description of the behavior of chgrp when encountering files greater than or equal to 2 Gbyte \(2^{31}\) bytes.

**Environment Variables**
See environ(5) for descriptions of the following environment variables that affect the execution of chgrp: LC_CTYPE, LC_MESSAGES, and NLSPATH.

**Exit Status**
The following exit values are returned:

- **0** The utility executed successfully and all requested changes were made.
- **>0** An error occurred.

**Files**
/etc/group group file

**Attributes**
See attributes(5) for descriptions of the following attributes:

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled (see NOTES)</td>
</tr>
</tbody>
</table>

**See Also**
chgrp(1), chmod(1), id(1M), chown(2), fpathconf(2), group(4), passwd(4), system(4), attributes(5), environ(5), largefile(5)

**Notes**
chgrp is CSI-enabled except for the group name.
chkey(1)

NAME

chkey – change user’s secure RPC key pair

SYNOPSIS

chkey [-p] [-s nisplus | nis | files | ldap] [-m <mechanism>]

DESCRIPTION

chkey is used to change a user’s secure RPC public key and secret key pair. chkey prompts for the old secure-rpc password and verifies that it is correct by decrypting the secret key. If the user has not already used keylogin(1) to decrypt and store the secret key with keyserv(1M), chkey registers the secret key with the local keyserv(1M) daemon. If the secure-rpc password does not match the login password, chkey prompts for the login password. chkey uses the login password to encrypt the user’s secret Diffie-Hellman (192 bit) cryptographic key. chkey can also encrypt other Diffie-Hellman keys for authentication mechanisms configured using nisauthconf(1M).

chkey ensures that the login password and the secure-rpc password(s) are kept the same, thus enabling password shadowing. See shadow(4).

The key pair can be stored in the /etc/publickey file (see publickey(4)), the NIS publickey map, or the NIS+ cred.org_dir table. If a new secret key is generated, it will be registered with the local keyserv(1M) daemon. However, only NIS+ can store Diffie-Hellman keys other than 192-bits.

Keys for specific mechanisms can be changed or reencrypted using the -m option followed by the authentication mechanism name. Multiple -m options can be used to change one or more keys. However, only mechanisms configured using nisauthconf(1M) can be changed with chkey.

If the source of the publickey is not specified with the -s option, chkey consults the publickey entry in the name service switch configuration file. See nsswitch.conf(4). If the publickey entry specifies one and only one source, then chkey will change the key in the specified name service. However, if multiple name services are listed, chkey can not decide which source to update and will display an error message. The user should specify the source explicitly with the -s option.

Non root users are not allowed to change their key pair in the files database.

OPTIONS

The following options are supported:

- -p Re-encrypt the existing secret key with the user’s login password.
- -s nisplus Update the NIS+ database.
- -s nis Update the NIS database.
- -s files Update the files database.
- -s ldap Update the LDAP database.
- -m <mechanism> Changes or re-encrypt the secret key for the specified mechanism.

User Commands  127
ATTRIBUTES See attributes(5) for descriptions of the following attributes:

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO keylogin(1), keylogout(1), keyserv(1M), newkey(1M), nisaddcred(1M), nisauthconf(1M), nsswitch.conf(4), publickey(4), shadow(4), attributes(5)

NOTES NIS+ might not be supported in future releases of the Solaris™ Operating Environment. Tools to aid the migration from NIS+ to LDAP are available in the Solaris 9 operating environment. For more information, visit http://www.sun.com/directory/nisplus/transition.html.
**NAME**
chmod – change the permissions mode of a file

**SYNOPSIS**

```plaintext
chmod [-fR] absolute-mode file...
chmod [-fR] symbolic-mode-list file...
```

**DESCRIPTION**
The `chmod` utility changes or assigns the mode of a file. The mode of a file specifies its permissions and other attributes. The mode may be absolute or symbolic.

**Absolute mode**
An absolute mode specification has the following format:

```plaintext
chmod [options] absolute-mode file ...where absolute-mode is specified using octal numbers nnnn defined as follows:
```

<table>
<thead>
<tr>
<th>n</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4000</td>
<td>Set user ID on execution.</td>
</tr>
<tr>
<td>20 # 0</td>
<td>Set group ID on execution if # is 7, 5, 3, or 1.</td>
</tr>
<tr>
<td></td>
<td>Enable mandatory locking if # is 6, 4, 2, or 0.</td>
</tr>
<tr>
<td></td>
<td>For directories, files are created with BSD semantics for propagation of the group ID. With this option, files and subdirectories created in the directory inherit the group ID of the directory, rather than of the current process. For directories, the set-gid bit may only be set or cleared by using symbolic mode.</td>
</tr>
<tr>
<td>1000</td>
<td>Turn on sticky bit. See <code>chmod(2)</code>.</td>
</tr>
<tr>
<td>0400</td>
<td>Allow read by owner.</td>
</tr>
<tr>
<td>0200</td>
<td>Allow write by owner.</td>
</tr>
<tr>
<td>0100</td>
<td>Allow execute (search in directory) by owner.</td>
</tr>
<tr>
<td>0070</td>
<td>Allow read, write, and execute (search) by owner.</td>
</tr>
<tr>
<td>0040</td>
<td>Allow read by group.</td>
</tr>
<tr>
<td>0020</td>
<td>Allow write by group.</td>
</tr>
<tr>
<td>0010</td>
<td>Allow execute (search in directory) by group.</td>
</tr>
<tr>
<td>0004</td>
<td>Allow read by others.</td>
</tr>
<tr>
<td>0002</td>
<td>Allow write by others.</td>
</tr>
<tr>
<td>0001</td>
<td>Allow execute (search in directory) by others.</td>
</tr>
<tr>
<td>0007</td>
<td>Allow read, write, and execute (search) by others.</td>
</tr>
</tbody>
</table>
For directories, the setgid bit cannot be set (or cleared) in absolute mode; it must be set (or cleared) in symbolic mode using g+s (or g-s).

**Symbolic mode**

A symbolic mode specification has the following format:

`chmod [options] symbolic-mode-list file . . .` where `symbolic-mode-list` is a comma-separated list (with no intervening whitespace) of symbolic mode expressions of the form:

```
[who] operator [permissions]
```

Operations are performed in the order given. Multiple `permissions` letters following a single operator cause the corresponding operations to be performed simultaneously.

who

zero or more of the characters u, g, o, and a specifying whose permissions are to be changed or assigned:

- u user’s permissions
- g group’s permissions
- o others’ permissions
- a all permissions (user, group, and other)

If `who` is omitted, it defaults to a, but the setting of the file mode creation mask (see `umask` in `sh(1)` or `csh(1)` for more information) is taken into account. When `who` is omitted, `chmod` will not override the restrictions of your user mask.

operator

either +, −, or =, signifying how permissions are to be changed:

- + Add permissions.
  
  If `permissions` is omitted, nothing is added.
  
  If `who` is omitted, add the file mode bits represented by `permissions, except for the those with corresponding bits in the file mode creation mask.
  
  If `who` is present, add the file mode bits represented by the `permissions`.

- − Take away permissions.
  
  If `permissions` is omitted, do nothing.
  
  If `who` is omitted, clear the file mode bits represented by `permissions, except for those with corresponding bits in the file mode creation mask.
  
  If `who` is present, clear the file mode bits represented by `permissions`. 
chmod(1)

= Assign permissions absolutely.

If who is omitted, clear all file mode bits; if who is present, clear the file mode bits represented by who.

If permissions is omitted, do nothing else.

If who is omitted, add the file mode bits represented by permissions, except for the those with corresponding bits in the file mode creation mask.

If who is present, add the file mode bits represented by permissions.

Unlike other symbolic operations, = has an absolute effect in that it resets all other bits represented by who. Omitting permissions is useful only with = to take away all permissions.

permission any compatible combination of the following letters:

l mandatory locking
r read permission
s user or group set-ID
t sticky bit
w write permission
x execute permission
X execute permission if the file is a directory or if there is execute permission for one of the other user classes
u,g,o indicate that permission is to be taken from the current user, group or other mode respectively.

Permissions to a file may vary depending on your user identification number (UID) or group identification number (GID). Permissions are described in three sequences each having three characters:

<table>
<thead>
<tr>
<th>User</th>
<th>Group</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>rwx</td>
<td>rwx</td>
<td>rwx</td>
</tr>
</tbody>
</table>

This example (user, group, and others all have permission to read, write, and execute a given file) demonstrates two categories for granting permissions: the access class and the permissions themselves.
The letter s is only meaningful with u or g, and t only works with u.

Mandatory file and record locking (1) refers to a file’s ability to have its reading or writing permissions locked while a program is accessing that file.

In a directory which has the set-group-ID bit set (reflected as either ----s--- or ----l--- in the output of ‘ls -ld’), files and subdirectories are created with the group-ID of the parent directory—not that of current process.

It is not possible to permit group execution and enable a file to be locked on execution at the same time. In addition, it is not possible to turn on the set-group-ID bit and enable a file to be locked on execution at the same time. The following examples, therefore, are invalid and elicit error messages:

```bash
chmod g+x,+l file
chmod g+s,+l file
```

Only the owner of a file or directory (or the super-user) may change that file’s or directory’s mode. Only the super-user may set the sticky bit on a non-directory file. If you are not super-user, chmod will mask the sticky-bit but will not return an error. In order to turn on a file’s set-group-ID bit, your own group ID must correspond to the file’s and group execution must be set.

**OPTIONS**

The following options are supported:

- `-f` Force. `chmod` will not complain if it fails to change the mode of a file.
- `-R` Recursively descends through directory arguments, setting the mode for each file as described above. When symbolic links are encountered, the mode of the target file is changed, but no recursion takes place.

**OPERANDS**

The following operands are supported:

- `absolute-mode`
- `symbolic-mode-list`

Represents the change to be made to the file mode bits of each file named by one of the `file` operands. See Absolute Mode and Symbolic Mode above in the DESCRIPTION section for more information.

- `file`

A path name of a file whose file mode bits are to be modified.

**USAGE**

See `largefile(5)` for the description of the behavior of `chmod` when encountering files greater than or equal to 2 Gbyte (2^31 bytes).

**EXAMPLES**

**EXAMPLE 1** Denying execute permission to everyone

```bash
example$ chmod a-x file
```
EXAMPLE 2 Allowing only read permission to everyone
example% chmod 444 file

EXAMPLE 3 Making a file readable and writable by the group and others
example% chmod go+rw file
example% chmod 066 file

EXAMPLE 4 Causing a file to be locked during access
example% chmod +l file

EXAMPLE 5 Allowing everyone to read, write, and execute the file and turn on the set group-ID
example% chmod a=rwx,g+s file
example% chmod 2777 file

ENVIRONMENT VARIABLES
See environ(5) for descriptions of the following environment variables that affect the execution of chmod: LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS
The following exit values are returned:
0 Successful completion.
>0 An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

SEE ALSO
getfacl(1), ls(1), setfacl(1), chmod(2), attributes(5), environ(5), largefile(5)

NOTES
Absolute changes do not work for the set-group-ID bit of a directory. You must use g+s or g-s.

chmod permits you to produce useless modes so long as they are not illegal (for instance, making a text file executable). chmod does not check the file type to see if mandatory locking is meaningful.

If the filesystem is mounted with the nosuid option, setuid execution is not allowed.
If you use `chmod` to change the file group owner permissions on a file with ACL entries, both the file group owner permissions and the ACL mask are changed to the new permissions. Be aware that the new ACL mask permissions may change the effective permissions for additional users and groups who have ACL entries on the file. Use the `getfacl(1)` command to make sure the appropriate permissions are set for all ACL entries.
NAME
chown – change file ownership

SYNOPSIS
chown [-fhR] owner [: group] file...

DESCRIPTION
The chown utility will set the user ID of the file named by each file to the user ID specified by owner, and, optionally, will set the group ID to that specified by group. If chown is invoked by other than the super-user, the set-user-ID bit is cleared.

Only the owner of a file (or the super-user) may change the owner of that file.

The operating system has a configuration option \{_POSIX_CHOWN_RESTRICTED\}, to restrict ownership changes. When this option is in effect the owner of the file is prevented from changing the owner ID of the file. Only the super-user can arbitrarily change owner IDs whether or not this option is in effect. To set this configuration option, include the following line in /etc/system:

```
set rstchown = 1
```

To disable this option, include the following line in /etc/system:

```
set rstchown = 0
```

\{_POSIX_CHOWN_RESTRICTED\} is enabled by default. See system(4) and fpathconf(2).

OPTIONS
The following options are supported:

- `f`
  Do not report errors.

- `h`
  If the file is a symbolic link, change the owner of the symbolic link. Without this option, the owner of the file referenced by the symbolic link is changed.

- `R`
  Recursive. chown descends through the directory, and any subdirectories, setting the ownership ID as it proceeds. When a symbolic link is encountered, the owner of the target file is changed (unless the -h option is specified), but no recursion takes place.

OPERANDS
The following operands are supported:

owner[ : group]
A user ID and optional group ID to be assigned to file. The owner portion of this operand must be a user name from the user database or a numeric user ID. Either specifies a user ID to be given to each file named by file. If a numeric owner exists in the user database as a user name, the user ID number associated with that user name will be used as the user ID. Similarly, if the group portion of this operand is present, it must be a group name from the group database or a numeric group ID. Either specifies a group ID to be given to each file. If a numeric group operand exists in the
group database as a group name, the group ID number associated with that group name will be used as the group ID.

`file` A path name of a file whose user ID is to be modified.

**USAGE** See `largefile(5)` for the description of the behavior of `chown` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

**EXAMPLES**

**EXAMPLE 1** Changing ownership of all files in the hierarchy

To change ownership of all files in the hierarchy, including symbolic links, but not the targets of the links:

```bash
example% chown -R -h owner[:group] file...
```

**ENVIRONMENT VARIABLES** See `environ(5)` for descriptions of the following environment variables that affect the execution of `chown`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

**EXIT STATUS** The following exit values are returned:

- `0` The utility executed successfully and all requested changes were made.
- `>0` An error occurred.

**FILES** `/etc/passwd` system password file

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled (see NOTES)</td>
</tr>
</tbody>
</table>

**SEE ALSO** `chgrp(1), chmod(1), chown(2), fpathconf(2), passwd(4), system(4), attributes(5), environ(5), largefile(5)`

**NOTES** `chown` is CSI-enabled except for the `owner` and `group` names.
chown(1B)

NAME  chown – change owner

SYNOPSIS  /usr/ucb/chown [-fR] owner [.group] filename...

DESCRIPTION  `chown` changes the owner of the filenames to `owner`. The owner may be either a decimal user ID (UID) or a login name found in the password file. An optional `group` may also be specified. The group may be either a decimal group ID (GID) or a group name found in the GID file.

Only the super-user can change owner, in order to simplify accounting procedures.

OPTIONS  

- **f**  Do not report errors.
- **R**  Recursively descend into directories setting the ownership of all files in each directory encountered. When symbolic links are encountered, their ownership is changed, but they are not traversed.

USAGE  See `largefile(5)` for the description of the behavior of `chown` when encountering files greater than or equal to 2 Gbyte (2³¹ bytes).

FILES  /etc/passwd password file

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

SEE ALSO  `chgrp(1), chown(2), group(4), passwd(4), attributes(5), largefile(5)`
The `ckdate` utility prompts a user and validates the response. It defines, among other things, a prompt message whose response should be a date, text for help and error messages, and a default value (which will be returned if the user responds with a RETURN). The user response must match the defined format for a date.

All messages are limited in length to 70 characters and are formatted automatically. Any white space used in the definition (including newline) is stripped. The `-W` option cancels the automatic formatting. When a tilde is placed at the beginning or end of a message definition, the default text will be inserted at that point, allowing both custom text and the default text to be displayed.

If the prompt, help or error message is not defined, the default message (as defined under `NOTES`) will be displayed.

Three visual tool modules are linked to the `ckdate` command. They are `errdate` (which formats and displays an error message), `helpdate` (which formats and displays a help message), and `valdate` (which validates a response). These modules should be used in conjunction with FML objects. In this instance, the FML object defines the prompt. When `format` is defined in the `errdate` and `helpdate` modules, the messages will describe the expected format.

The following options are supported:

- `-d default` Defines the default value as `default`. The default does not have to meet the format criteria.
- `-e error` Defines the error message as `error`.
- `-f format` Specifies the format against which the input will be verified. Possible formats and their definitions are:

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>%b</code></td>
<td>abbreviated month name (jan, feb, mar)</td>
</tr>
<tr>
<td><code>%B</code></td>
<td>full month name</td>
</tr>
<tr>
<td><code>%d</code></td>
<td>day of month (01 - 31)</td>
</tr>
<tr>
<td><code>%D</code></td>
<td>date as %m/%d/%y (the default format)</td>
</tr>
<tr>
<td><code>%e</code></td>
<td>day of month (1 - 31; single digits are preceded by a blank)</td>
</tr>
<tr>
<td><code>%h</code></td>
<td>abbreviated month name, identical to <code>%b%</code></td>
</tr>
<tr>
<td><code>%m</code></td>
<td>month number (01 - 12)</td>
</tr>
</tbody>
</table>
%y = year within century (for instance, 89)
%Y = year as CCYY (for instance, 1989)

-`h` help
  Defines the help messages as `help`.

-`k` pid
  Specifies that process ID `pid` is to be sent a signal if the user chooses to abort.

-`p` prompt
  Defines the prompt message as `prompt`.

-`Q`
  Specifies that quit will not be allowed as a valid response.

-`s` signal
  Specifies that the process ID `pid` defined with the `-k` option is to be sent signal `signal` when quit is chosen. If no signal is specified, SIGTERM is used.

-`W` width
  Specifies that prompt, help and error messages will be formatted to a line length of `width`.

**OPERANDS**

The following operand is supported:

`input` Input to be verified against format criteria.

**EXIT STATUS**

The following exit values are returned:

0  Successful execution.
1  EOF on input, or negative width on `-W` option, or usage error.
3  User termination (quit).
4  Garbled format argument.

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

**SEE ALSO**

`attributes(5)`

**NOTES**

The default prompt for `ckdate` is:

Enter the date [?q]:

The default error message is:

```
ERROR - Please enter a date. Format is <format>.
```

The default help message is:

```
Please enter a date. Format is <format>.
```
When the quit option is chosen (and allowed), q is returned along with the return code 3. The validate module will not produce any output. It returns zero for success and non-zero for failure.
ckgid prompts a user and validates the response. It defines, among other things, a prompt message whose response should be an existing group ID, text for help and error messages, and a default value (which will be returned if the user responds with a carriage return).

All messages are limited in length to 70 characters and are formatted automatically. Any white space used in the definition (including newline) is stripped. The `-W` option cancels the automatic formatting. When a tilde is placed at the beginning or end of a message definition, the default text will be inserted at that point, allowing both custom text and the default text to be displayed.

If the prompt, help or error message is not defined, the default message (as defined under NOTES) will be displayed.

Three visual tool modules are linked to the ckgid command. They are errgid (which formats and displays an error message), helpgid (which formats and displays a help message), and valgid (which validates a response). These modules should be used in conjunction with FML objects. In this instance, the FML object defines the prompt.

The following options are supported:

- `-d default`: Defines the default value as `default`. The default is not validated and so does not have to meet any criteria.
- `-e error`: Defines the error message as `error`.
- `-h help`: Defines the help messages as `help`.
- `-k pid`: Specifies that process ID `pid` is to be sent a signal if the user chooses to abort.
- `-m`: Displays a list of all groups when help is requested or when the user makes an error.
- `-p prompt`: Defines the prompt message as `prompt`.
- `-Q`: Specifies that quit will not be allowed as a valid response.
- `-s signal`: Specifies that the process ID `pid` defined with the `-k` option is to be sent signal `signal` when quit is chosen. If no signal is specified, SIGTERM is used.
ckgid(1)

-w width  Specifies that prompt, help and error messages will be formatted to a line length of width.

OPERANDS  The following operand is supported:
           input      Input to be verified against /etc/group.

EXIT STATUS The following exit values are returned:
           0        Successful execution.
           1        EOF on input, or negative width on -w option, or usage error.
           3        User termination (quit).

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO  attributes(5)

NOTES  The default prompt for ckgid is:
        Enter the name of an existing group [?,q]:

The default error message is:
ERROR: Please enter one of the following group names: [List]

If the -m option of ckgid is used, a list of valid groups is displayed here.

The default help message is:
ERROR: Please enter one of the following group names: [List]

If the -m option of ckgid is used, a list of valid groups is displayed here.

When the quit option is chosen (and allowed), q is returned along with the return code 3. The valgid module will not produce any output. It returns 0 for success and non-zero for failure.
ckint, errint, helpint, valint – display a prompt; verify and return an integer value

SYNOPSIS

       [-p prompt] [-k pid [-s signal]]
/usr/sadm/bin/errint [-W width] [-b base] [-e error]
/usr/sadm/bin/helpint [-W width] [-b base] [-h help]
/usr/sadm/bin/valint [-b base] input

DESCRIPTION

The ckint utility prompts a user, then validates the response. It defines, among other things, a prompt message whose response should be an integer, text for help and error messages, and a default value (which will be returned if the user responds with a carriage return).

All messages are limited in length to 70 characters and are formatted automatically. Any white space used in the definition (including newline) is stripped. The -W option cancels the automatic formatting. When a tilde is placed at the beginning or end of a message definition, the default text will be inserted at that point, allowing both custom text and the default text to be displayed.

If the prompt, help or error message is not defined, the default message (as defined under NOTES) will be displayed.

Three visual tool modules are linked to the ckint command. They are errint (which formats and displays an error message), helpint (which formats and displays a help message), and valint (which validates a response). These modules should be used in conjunction with FML objects. In this instance, the FML object defines the prompt. When base is defined in the errint and helpint modules, the messages will include the expected base of the input.

OPTIONS

The following options are supported:

- b base        Defines the base for input. Must be 2 to 36, default is 10.
- d default     Defines the default value as default. The default is not validated and so does not have to meet any criteria.
- e error       Defines the error message as error.
- h help        Defines the help messages as help.
- k pid         Specifies that process ID pid is to be sent a signal if the user chooses to abort.
- p prompt      Defines the prompt message as prompt.
- Q             Specifies that quit will not be allowed as a valid response.
- s signal      Specifies that the process ID pid defined with the -k option is to be sent signal signal when quit is chosen. If no signal is specified, SIGTERM is used.
ckint(1)

-\n width  Specifies that prompt, help and error messages will be formatted to a line length of width.

OPERANDS  The following operand is supported:

input  Input to be verified against base criterion.

EXIT STATUS  The following exit values are returned:

0  Successful execution.
1  EOF on input, or negative width on -\n option, or usage error.
3  User termination (quit).

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO  attributes(5)

NOTES  The default base 10 prompt for ckint is:

Enter an integer [?,q]:

The default base 10 error message is:

ERROR - Please enter an integer.

The default base 10 help message is:

Please enter an integer.

The messages are changed from "integer" to "base base integer" if the base is set to a number other than 10.

When the quit option is chosen (and allowed), q is returned along with the return code 3. The valint module will not produce any output. It returns 0 for success and non-zero for failure.
ckitem, erritem, helpitem – build a menu; prompt for and return a menu item


/usr/sadm/bin/erritem [-W width] [-e error] [ choice [...]]

/usr/sadm/bin/helpitem [-W width] [-h help] [ choice [...]]

DESCRIPTION
The ckitem utility builds a menu and prompts the user to choose one item from a menu of items. It then verifies the response. Options for this command define, among other things, a prompt message whose response will be a menu item, text for help and error messages, and a default value (which will be returned if the user responds with a carriage return).

By default, the menu is formatted so that each item is prepended by a number and is printed in columns across the terminal. Column length is determined by the longest choice. Items are alphabetized.

All messages are limited in length to 70 characters and are formatted automatically. Any white space used in the definition (including newline) is stripped. The -W option cancels the automatic formatting. When a tilde is placed at the beginning or end of a message definition, the default text will be inserted at that point, allowing both custom text and the default text to be displayed.

If the prompt, help or error message is not defined, the default message (as defined under NOTES) will be displayed.

Two visual tool modules are linked to the ckitem command. They are erritem (which formats and displays an error message) and helpitem (which formats and displays a help message). These modules should be used in conjunction with FML objects. In this instance, the FML object defines the prompt. When choice is defined in these modules, the messages will describe the available menu choice (or choices).

OPTIONS
The following options are supported:

-default
Define the default value as default. The default is not validated and so does not have to meet any criteria.

-error
Define the error message as error.

-filename
Define a file, filename, which contains a list of menu items to be displayed. (The format of this file is: token<tab>description. Lines beginning with a pound sign (#) are designated as comments and ignored.)

-help
Define the help messages as help.

-invis
Define invisible menu choices (those which will not be printed in the menu). (For example, “all” used as an invisible choice would mean it is a legal option but does not appear in the menu. Any
number of invisible choices may be defined.) Invisible choices should be made known to a user either in the prompt or in a help message.

-k pid Specify that the process ID pid is to be sent a signal if the user chooses to abort.

-l label Define a label, label, to print above the menu.

-m max Define the maximum number of menu choices that the user can choose. The default is 1.

-n Specify that menu items should not be displayed in alphabetical order.

-o Specify that only one menu token will be returned.

-p prompt Define the prompt message as prompt.

-Q Specify that quit will not be allowed as a valid response.

-s signal Specify that process ID pid defined with the -k option is to be sent signal signal when quit is chosen. If no signal is specified, SIGTERM is used.

-u Specify that menu items should be displayed as an unnumbered list.

-W width Specify that prompt, help and error messages will be formatted to a line length of width.

OPERANDS The following operand is supported:
choice Define menu items. Items should be separated by white space or newline.

EXIT STATUS The following exit values are returned:
0 Successful execution.
1 EOF on input, or negative width on -W option, or inability to open file on -f option, or usage error.
3 User termination (quit).
4 No choices from which to choose.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO attributes(5)
The user may input the number of the menu item if choices are numbered or as much of the string required for a unique identification of the item. Long menus are paged with 10 items per page.

When menu entries are defined both in a file (by using the -f option) and also on the command line, they are usually combined alphabetically. However, if the -n option is used to suppress alphabetical ordering, then the entries defined in the file are shown first, followed by the options defined on the command line.

The default prompt for ckitem is:

Enter selection [?,??,q]:

One question mark will give a help message and then redisplay the prompt. Two question marks will give a help message and then redisplay the menu label, the menu and the prompt.

The default error message if you typed a number is:

ERROR: Bad numeric choice specification

The default error message if you typed a string is:

ERROR: Entry does not match available menu selection. Enter the number of the menu item you wish to select, the token which is associated with the menu item, or a partial string which uniquely identifies the token for the menu item. Enter ?? to reprint the menu.

The default help message is:

Enter the number of the menu item you wish to select, the token which is associated with the menu item, or a partial string which uniquely identifies the token for the menu item. Enter ? to reprint the menu.

When the quit option is chosen (and allowed), q is returned along with the return code 3.
ckkeywd(1)

**NAME**
ckkeywd – prompts for and validates a keyword

**SYNOPSIS**
   [-k pid] [-s signal] keyword [ .. ]

**DESCRIPTION**
ckkeywd prompts a user and validates the response. It defines, among other things, a prompt message whose response should be one of a list of keywords, text for help and error messages, and a default value (which will be returned if the user responds with a carriage return). The answer returned from this command must match one of the defined list of keywords.

All messages are limited in length to 70 characters and are formatted automatically. Any white space used in the definition (including newline) is stripped. The -W option cancels the automatic formatting. When a tilde is placed at the beginning or end of a message definition, the default text will be inserted at that point, allowing both custom text and the default text to be displayed.

If the prompt, help or error message is not defined, the default message (as defined under NOTES) will be displayed.

**OPTIONS**
The following options are supported:

- `-d default`  Defines the default value as default. The default is not validated and so does not have to meet any criteria.
- `-e error`  Defines the error message as error.
- `-h help`  Defines the help messages as help.
- `-k pid`  Specifies that process ID pid is to be sent a signal if the user chooses to abort.
- `-p prompt`  Defines the prompt message as prompt.
- `-Q`  Specifies that quit will not be allowed as a valid response.
- `-s signal`  Specifies that the process ID pid defined with the -k option is to be sent signal signal when quit is chosen. If no signal is specified, SIGTERM is used.
- `-W width`  Specifies that prompt, help and error messages will be formatted to a line length of width.

**OPERANDS**
The following operand is supported:

- `keyword` Defines the keyword, or list of keywords, against which the answer will be verified.

**EXIT STATUS**
The following exit values are returned:

- `0`  Successful execution.
- `1` EOF on input, or negative width on -W option, or no keywords from which to choose, or usage error.
User Commands 149

User termination (quit).

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO
attributes(5)

NOTES
The default prompt for ckkeywd is:

Enter appropriate value [keyword, [ . . . ], ?, q]:

The default error message is:

ERROR: Please enter one of the following keywords: keyword, [ . . . ], q

The default help message is:

keyword, [ . . . ], q

When the quit option is chosen (and allowed), q is returned along with the return code 3.
ckpath(1)

NAME
ckpath, errpath, helppath, valpath – display a prompt; verify and return a pathname

SYNOPSIS
ckpath [-Q] [-W width] [-a | l] [-b | c | f | y] [-n [o | z]]
       [-s signal]]
/usr/sadm/bin/errpath [-W width] [-a | l] [-b | c | f | y] [-n [o |
       z]] [-rtwx] [-e error]
/usr/sadm/bin/helppath [-W width] [-a | l] [-b | c | f | y] [-n [o |
       z]] [-rtwx] [-h help]
/usr/sadm/bin/valpath [-a | l] [-b | c | f | y] [-n [o | z]]
       [-rtwx] input

DESCRIPTION
The ckpath utility prompts a user and validates the response. It defines, among other
things, a prompt message whose response should be a pathname, text for help and
error messages, and a default value (which is returned if the user responds with a
RETURN).

The pathname must obey the criteria specified by the first group of options. If no
criteria is defined, the pathname must be for a normal file that does not yet exist. If
neither -a (absolute) or -l (relative) is given, then either is assumed to be valid.

All messages are limited in length to 79 characters and are formatted automatically.
Tabs and newlines are removed after a single white space character in a message
definition, but spaces are not removed. When a tilde is placed at the beginning or end
of a message definition, the default text is inserted at that point, allowing both custom
text and the default text to be displayed.

If the prompt, help or error message is not defined, the default message (as defined
under EXAMPLES) is displayed.

Three visual tool modules are linked to the ckpath command. They are errpath
(which formats and displays an error message on the standard output), helppath
(which formats and displays a help message on the standard output), and valpath
(which validates a response). These modules should be used in conjunction with
Framed Access Command Environment (FACE) objects. In this instance, the FACE
object defines the prompt.

OPTIONS
The following options are supported:
- a Pathname must be an absolute path.
- b Pathname must be a block special file.
- c Pathname must be a character special file.
- d default Defines the default value as default. The default is not validated
and so does not have to meet any criteria.
- e error Defines the error message as error.
- f Pathname must be a regular file.
ckpath(1)

-h help  Defines the help message as help.
-k pid   Specifies that process ID pid is to be sent a signal if the user
         chooses to quit.
-l       Pathname must be a relative path.
-n       Pathname must not exist (must be new).
-o       Pathname must exist (must be old).
-p prompt  Defines the prompt message as prompt.
-q       Specifies that quit is not allowed as a valid response.
-r       Pathname must be readable.
-s signal Specifies that the process ID pid defined with the -k option is to be
            sent signal signal when quit is chosen. If no signal is specified,
            SIGTERM is used.
-t       Pathname must be creatable (touchable). Pathname will be created
         if it does not already exist.
-w       Pathname must be writable.
-w width Specify that prompt, help and error messages be formatted to a
         line length of width.
-x       Pathname must be executable.
-y       Pathname must be a directory.
-z       Pathname must have a file having a size greater than zero bytes.

OPERANDS
The following operand is supported:

input    Input to be verified against validation options.

EXAMPLES
The text of the default messages for ckpath depends upon the criteria options that
have been used.

EXAMPLE 1 Default prompt
An example default prompt for ckpath (using the -a option) is:

example$ ckpath -a
Enter an absolute pathname [?,q]

EXAMPLE 2 Default error message
An example default error message (using the -a option) is:

example$ /usr/sadm/bin/errpath -a
ERROR: A pathname is a filename, optionally preceded by parent directories.
The pathname you enter: - must begin with a slash (/)
EXAMPLE 3 Default help message

An example default help message (using the -a option) is:

```
example% /usr/sadm/bin/helppath -a
```

A pathname is a filename, optionally preceded by parent directories.
The pathname you enter: - must begin with a slash (/)

EXAMPLE 4 The quit option

When the quit option is chosen (and allowed), q is returned along with the return code
3. Quit input gets a trailing newline.

EXAMPLE 5 Using the valpath module

The valpath module will produce a usage message on stderr. It returns 0 for success
and non-zero for failure.

```
exa[168x712]mple% /usr/sadm/bin/valpath
usage: valpath [-a|-l][b|c|f|y][n|o|z]rtwx] input
```

EXIT STATUS

The following exit values are returned:

0 Successful execution.
1 EOF on input, or negative width on -W option, or usage error.
2 Mutually exclusive options.
3 User termination (quit).
4 Mutually exclusive options.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO

face(1), signal(3HEAD), attributes(5)
ckrange, errange, helprange, valrange -- prompts for and validates an integer

/usr/sadm/bin/valrange [-l lower] [-u upper] [-b base] input

DESCRIPTION

The ckrange utility prompts a user for an integer between a specified range and determines whether this response is valid. It defines, among other things, a prompt message whose response should be an integer in the range specified, text for help and error messages, and a default value (which is returned if the user responds with a RETURN).

This command also defines a range for valid input. If either the lower or upper limit is left undefined, then the range is bounded on only one end.

All messages are limited in length to 79 characters and are formatted automatically. Tabs and newlines are removed after a single whitespace character in a message definition, but spaces are not removed. When a tilde is placed at the beginning or end of a message definition, the default text will be inserted at that point, allowing both custom text and the default text to be displayed.

If the prompt, help or error message is not defined, the default message (as defined under EXAMPLES) is displayed.

Three visual tool modules are linked to the ckrange command. They are errange (which formats and displays an error message on the standard output), helprange (which formats and displays a help message on the standard output), and valrange (which validates a response). These modules should be used in conjunction with Framed Access Command Environment (FACE) objects. In this instance, the FACE object defines the prompt.

Note: Negative "input" arguments confuse getopt in valrange. By inserting a "-" before the argument, getopt processing will stop. See getopt(1) and intro(1) about getopt parameter handling. getopt is used to parse positional parameters and to check for legal options.

OPTIONS

The following options are supported:

-**b** base
  Defines the base for input. Must be 2 to 36, default is 10. Base conversion uses strtol(3C). Output is always base 10.

-**d** default
  Defines the default value as **default**. **default** is converted using strtol(3C) in the desired base. Any characters invalid in the specified base will terminate the strtol conversion without error.
cKrange(1)

- **-e error** Defines the error message as `error`.
- **-h help** Defines the help message as `help`.
- **-k pid** Specifies that process ID `pid` is to be sent a signal if the user chooses to quit.
- **-l lower** Defines the lower limit of the range as `lower`. Default is the machine's largest negative long.
- **-p prompt** Defines the prompt message as `prompt`.
- **-Q** Specifies that quit will not be allowed as a valid response.
- **-a signal** Specifies that the process ID `pid` defined with the `-k` option is to be sent signal `signal` when quit is chosen. If no signal is specified, `SIGTERM` is used.
- **-u upper** Defines the upper limit of the range as `upper`. Default is the machine's largest positive long.
- **-w width** Specifies that prompt, help and error messages will be formatted to a line length of `width`.

**OPERANDS**
The following operand is supported:

*input* Input to be verified against upper and lower limits and base.

**EXAMPLES**

**EXAMPLE 1** Default base 10 prompt

The default base 10 prompt for `ckrange` is:

```
example$ ckrange
Enter an integer between lower_bound and upper_bound [lower_bound-upper_bound, ?, q]:
```

**EXAMPLE 2** Default base 10 error message

The default base 10 error message is:

```
example$ /usr/sadm/bin/errange
ERROR: Please enter an integer between lower_bound \ and upper_bound.
```

**EXAMPLE 3** Default base 10 help message

The default base 10 help message is:

```
example$ /usr/sadm/bin/helprange
Please enter an integer between lower_bound and upper_bound.
```
EXAMPLE 4 Changing messages for a base other than 10

The messages are changed from “integer” to “base base integer” if the base is set to a number other than 10. For example,

```
example% /usr/sadm/bin/helprange -b 36
```

EXAMPLE 5 Using the quit option

When the quit option is chosen (and allowed), q is returned along with the return code 3. Quit input gets a trailing newline.

EXAMPLE 6 Using the valrange module

The valrange module will produce a usage message on stderr. It returns 0 for success and non-zero for failure.

```
example% /usr/sadm/bin/valrange
usage: valrange [-l lower] [-u upper] [-b base] input
```

EXIT STATUS The following exit values are returned:

0 Successful execution.
1 EOF on input, or negative width on -W option, or usage error.
2 Usage error.
3 User termination (quit).

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO intro(1), face(1), getopt(1), strtol(3C), attributes(5), signal(3HEAD)
ckstr(1)

NAME
ckstr, errstr, helpstr, valstr – display a prompt; verify and return a string answer

SYNOPSIS
ckstr [-Q] [-W width] [ [-r regexp] [...] [-l length] [-d default]
          [-h help] [-e error] [-p prompt] [-k pid [- s signal]]
/usr/sadm/bin/errstr [-W width] [-e error] [-l length] [ [-r regexp]
                      [...]]
/usr/sadm/bin/helpstr [-W width] [-h help] [-l length] [ [-r regexp]
                       [...]]
/usr/sadm/bin/valstr [-l length] [ [-r regexp] [...] input

DESCRIPTION
The ckstr utility prompts a user and validates the response. It defines, among other things, a prompt message whose response should be a string, text for help and error messages, and a default value (which are returned if the user responds with a RETURN).

The answer returned from this command must match the defined regular expression and be no longer than the length specified. If no regular expression is given, valid input must be a string with a length less than or equal to the length defined with no internal, leading or trailing white space. If no length is defined, the length is not checked.

All messages are limited in length to 79 characters and are formatted automatically. Tabs and newlines are removed after a single white space character in a message definition, but spaces are not removed. When a tilde is placed at the beginning or end of a message definition, the default text will be inserted at that point, allowing both custom text and the default text to be displayed.

If the prompt, help or error message is not defined, the default message (as defined under EXAMPLES) is displayed.

Three visual tool modules are linked to the ckstr command. They are errstr (which formats and displays an error message on the standard output), helpstr (which formats and displays a help message on the standard output), and valstr (which validates a response). These modules should be used in conjunction with Framed Access Command Environment (FACE) objects. In this instance, the FACE object defines the prompt.

OPTIONS
The following options are supported:

- **-d default**
  Defines the default value as default. The default is not validated and so does not have to meet any criteria.

- **-e error**
  Defines the error message as error.

- **-h help**
  Defines the help message as help.

- **-k pid**
  Specifies that process ID pid is to be sent a signal if the user chooses to quit.

- **-l length**
  Specifies the maximum length of the input.
ckstr(1)

OPERANDS

The following operand is supported:

input Input to be verified against format length and/or regular expression criteria.

EXAMPLES

EXAMPLE 1 Default prompt

The default prompt for ckstr is:

eexample: ckstr
Enter an appropriate value [?,q]:

EXAMPLE 2 Default error message

The default error message is dependent upon the type of validation involved. The user will be told either that the length or the pattern matching failed. The default error message is:

eexample: /usr/sadm/bin/errstr
ERROR: Please enter a string which contains no embedded, leading or trailing spaces or tabs.

EXAMPLE 3 Default help message

The default help message is also dependent upon the type of validation involved. If a regular expression has been defined, the message is:

eexample: /usr/sadm/bin/helpstr -r regexp
Please enter a string which matches the following pattern:
regexp

Other messages define the length requirement and the definition of a string.

EXAMPLE 4 Using the quit option

When the quit option is chosen (and allowed), q is returned along with the return code 3. Quit input gets a trailing newline.
EXAMPLE 5 Using the valstr module

The valstr module will produce a usage message on stderr. It returns 0 for success and non-zero for failure.

```
example$ /usr/sadm/bin/valstr
usage: valstr [-l length] [[-r regexp] [ . . . ]] input
```

EXIT STATUS

The following exit values are returned:

<table>
<thead>
<tr>
<th>Exit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful execution.</td>
</tr>
<tr>
<td>1</td>
<td>EOF on input, or negative width on -W option, or usage error.</td>
</tr>
<tr>
<td>2</td>
<td>Invalid regular expression.</td>
</tr>
<tr>
<td>3</td>
<td>User termination (quit).</td>
</tr>
</tbody>
</table>

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>Attribute Type</th>
<th>Attribute Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO

face(1), signal(3HEAD), attributes(5)
cksum(1)

NAME  cksum – write file checksums and sizes

SYNOPSIS  cksum [file...]

DESCRIPTION  The cksum command calculates and writes to standard output a cyclic redundancy check (CRC) for each input file, and also writes to standard output the number of octets in each file.

For each file processed successfully, cksum will write in the following format:

"%u %d %s\n" <checksum>, <# of octets>, <path name>

If no file operand was specified, the path name and its leading space will be omitted.

The CRC used is based on the polynomial used for CRC error checking in the referenced Ethernet standard.

The encoding for the CRC checksum is defined by the generating polynomial:

\[ G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1 \]

Mathematically, the CRC value corresponding to a given file is defined by the following procedure:

1. The \(n\) bits to be evaluated are considered to be the coefficients of a mod 2 polynomial \(M(x)\) of degree \(n–1\). These \(n\) bits are the bits from the file, with the most significant bit being the most significant bit of the first octet of the file and the last bit being the least significant bit of the last octet, padded with zero bits (if necessary) to achieve an integral number of octets, followed by one or more octets representing the length of the file as a binary value, least significant octet first. The smallest number of octets capable of representing this integer is used.

2. \(M(x)\) is multiplied by \(x^{32}\) (that is, shifted left 32 bits) and divided by \(G(x)\) using mod 2 division, producing a remainder \(R(x)\) of degree \(\leq 31\).

3. The coefficients of \(R(x)\) are considered to be a 32-bit sequence.

4. The bit sequence is complemented and the result is the CRC.

OPERANDS  The following operand is supported:

file  A path name of a file to be checked. If no file operands are specified, the standard input is used.

USAGE  The cksum command is typically used to quickly compare a suspect file against a trusted version of the same, such as to ensure that files transmitted over noisy media arrive intact. However, this comparison cannot be considered cryptographically secure. The chances of a damaged file producing the same CRC as the original are astronomically small; deliberate deception is difficult, but probably not impossible.
Although input files to `cksum` can be any type, the results need not be what would be expected on character special device files. Since this document does not specify the block size used when doing input, checksums of character special files need not process all of the data in those files.

The algorithm is expressed in terms of a bitstream divided into octets. If a file is transmitted between two systems and undergoes any data transformation (such as moving 8-bit characters into 9-bit bytes or changing “Little Endian” byte ordering to “Big Endian”), identical CRC values cannot be expected. Implementations performing such transformations may extend `cksum` to handle such situations.

See `largefile(5)` for the description of the behavior of `cksum` when encountering files greater than or equal to 2 Gbyte ($2^{31}$ bytes).

**ENVIRONMENT VARIABLES**

See `environ(5)` for descriptions of the following environment variables that affect the execution of `cksum`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

**EXIT STATUS**

The following exit values are returned:

<table>
<thead>
<tr>
<th>Exit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>All files were processed successfully.</td>
</tr>
<tr>
<td>&gt;0</td>
<td>An error occurred.</td>
</tr>
</tbody>
</table>

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>Attribute Type</th>
<th>Attribute Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsus</td>
</tr>
</tbody>
</table>

**SEE ALSO**

`sum(1), attributes(5), environ(5), largefile(5)`
cktime utility prompts a user and validates the response. It defines, among other things, a prompt message whose response should be a time, text for help and error messages, and a default value (which is returned if the user responds with a RETURN). The user response must match the defined format for the time of day.

All messages are limited in length to 70 characters and are formatted automatically. Any white space used in the definition (including NEWLINE) is stripped. The `-W` option cancels the automatic formatting. When a tilde is placed at the beginning or end of a message definition, the default text is inserted at that point, allowing both custom text and the default text to be displayed.

If the prompt, help or error message is not defined, the default message (as defined under NOTES) is displayed.

Three visual tool modules are linked to the cktime command. They are errtime (which formats and displays an error message), helptime (which formats and displays a help message), and valtime (which validates a response). These modules should be used in conjunction with FML objects. In this instance, the FML object defines the prompt. When format is defined in the errtime and helptime modules, the messages will describe the expected format.

The following options are supported:

- `-d default` Defines the default value as `default`. The default is not validated and so does not have to meet any criteria.
- `-e error` Defines the error message as `error`.
- `-f format` Specifies the format against which the input will be verified.

Possible formats and their definitions are:

- `%H` = hour (00 - 23)
- `%I` = hour (00 - 12)
- `%M` = minute (00 - 59)
- `%p` = ante meridian or post meridian
- `%r` = time as `%I:%M:%S %p`
- `%R` = time as `%H:%M` (the default format)
- `%S` = seconds (00 - 59)
- `%T` = time as `%H:%M:%S`

- `-h help` Defines the help messages as `help`.
cktime(1)

-\k pid Specifies that process ID pid is to be sent a signal if the user chooses to abort.
-\p prompt Defines the prompt message as prompt.
-\Q Specifies that quit will not be allowed as a valid response.
-\a signal Specifies that the process ID pid defined with the -\k option is to be sent signal signal when quit is chosen. If no signal is specified, SIGTERM is used.
-\w width Specifies that prompt, help and error messages will be formatted to a line length of width.

OPERANDS The following operand is supported:

input Input to be verified against format criteria.

EXIT STATUS The following exit values are returned:
0 Successful execution.
1 EOF on input, or negative width on -\w option, or usage error.
3 User termination (quit).
4 Garbled format argument.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO attributes(5)

NOTES The default prompt for ckt ime is:
Enter a time of day [?,q]:

The default error message is:
ERROR: Please enter the time of day. Format is <format>.

The default help message is:
Please enter the time of day. Format is <format>.

When the quit option is chosen (and allowed), q is returned along with the return code 3. The valtime module will not produce any output. It returns 0 for success and non-zero for failure.
NAME
ckuid, erruid, helpuid, valuid – prompts for and validates a user ID

SYNOPSIS
       [-k pid] [-s signal]
/usr/sadm/bin/erruid [-W width] [-e error]
/usr/sadm/bin/helpuid [-W width] [-m] [-h help]
/usr/sadm/bin/valuid input

DESCRIPTION
The ckuid utility prompts a user and validates the response. It defines, among other
gthings, a prompt message whose response should be an existing user ID, text for help
and error messages, and a default value (which are returned if the user responds with
a RETURN).

All messages are limited in length to 70 characters and are formatted automatically.
Any white space used in the definition (including NEWLINE) is stripped. The -W
option cancels the automatic formatting. When a tilde is placed at the beginning or
end of a message definition, the default text is inserted at that point, allowing both
custom text and the default text to be displayed.

If the prompt, help or error message is not defined, the default message (as defined
under NOTES) is displayed.

Three visual tool modules are linked to the ckuid command. They are erruid
(which formats and displays an error message), helpuid (which formats and displays
a help message), and valuid (which validates a response). These modules should be
used in conjunction with FML objects. In this instance, the FML object defines the
prompt.

OPTIONS
The following options are supported:

- d default
  Defines the default value as default. The default is not validated
  and so does not have to meet any criteria.

- e error
  Defines the error message as error.

- h help
  Defines the help messages as help.

- k pid
  Specifies that process ID pid is to be sent a signal if the user
  chooses to abort.

- m
  Displays a list of all logins when help is requested or when the
  user makes an error.

- p prompt
  Defines the prompt message as prompt.

- Q
  Specifies that quit will not be allowed as a valid response.

- s signal
  Specifies that the process ID pid defined with the -k option is to be
  sent signal signal when quit is chosen. If no signal is specified,
  SIGTERM is used.
ckuid(1)

-\n width Specifies that prompt, help and error messages will be formatted to a line length of width.

OPERANDS
The following operand is supported:

input Input to be verified against /etc/passwd.

EXIT STATUS
The following exit values are returned:

0 Successful execution.
1 EOF on input, or negative width on -\n option, or usage error.
2 Usage error.
3 User termination (quit).

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO
attributes(5)

NOTES
The default prompt for ckuid is:

Enter the login name of an existing user [?,q]:

The default error message is:

ERROR - Please enter the login name of an existing user.

If the -m option is used, the default error message is:

ERROR: Please enter one of the following login names: <List>

The default help message is:

Please enter the login name of an existing user.

If the -m option is used, the default help message is:

Please enter one of the following login names: <List>

When the quit option is chosen (and allowed), q is returned along with the return code 3. The valuidd module will not produce any output. It returns 0 for success and non-zero for failure.
ckyorn prompts a user and validates the response. It defines, among other things, a prompt message for a yes or no answer, text for help and error messages, and a default value (which is returned if the user responds with a RETURN).

All messages are limited in length to 70 characters and are formatted automatically. Any white space used in the definition (including newline) is stripped. The -W option cancels the automatic formatting. When a tilde is placed at the beginning or end of a message definition, the default text is inserted at that point, allowing both custom text and the default text to be displayed.

If the prompt, help or error message is not defined, the default message (as defined under NOTES) is displayed.

Three visual tool modules are linked to the ckyorn command. They are erryorn (which formats and displays an error message), helpyorn (which formats and displays a help message), and valyorn (which validates a response). These modules should be used in conjunction with FACE objects. In this instance, the FACE object defines the prompt.

The following options are supported:

- `-d default` Defines the default value as `default`. The default is not validated and so does not have to meet any criteria.
- `-e error` Defines the error message as `error`.
- `-h help` Defines the help messages as `help`.
- `-k pid` Specifies that process ID `pid` is to be sent a signal if the user chooses to abort.
- `-p prompt` Defines the prompt message as `prompt`.
- `-Q` Specifies that `quit` will not be allowed as a valid response.
- `-s signal` Specifies that the process ID `pid` defined with the `-k` option is to be sent signal `signal` when `quit` is chosen. If no signal is specified, `SIGTERM` is used.
- `-W width` Specifies that prompt, help and error messages will be formatted to a line length of `width`.

The following operand is supported:

ckyorn(1)

**input**
Input to be verified as y, yes, or n, no (in any combination of upper- and lower-case letters).

**EXIT STATUS**
The following exit values are returned:

0  Successful execution.
1  EOF on input, or negative width on -W option, or usage error.
2  Usage error.
3  User termination (quit).

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

**SEE ALSO**
attributes(5)

**NOTES**
The default prompt for ckyorn is:

Yes or No [y,n,?,q]:

The default error message is:

ERROR - Please enter yes or no.

The default help message is:

To respond in the affirmative, enter y, yes, Y, or YES.
To respond in the negative, enter n, no, N, or NO.

When the quit option is chosen (and allowed), q is returned along with the return code 3. The valyorn module will not produce any output. It returns 0 for success and non-zero for failure.
clear – clear the terminal screen

SYNOPSIS

clear

DESCRIPTION
clear clears your screen if this is possible. It looks in the environment for the
terminal type and then in the terminfo database to figure out how to clear the screen.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO
attributes(5)
cmp(1)

NAME    cmp – compare two files
SYNOPSIS cmp  [-l]  [-s] file1 file2  [skip1]  [skip2]
DESCRIPTION The cmp utility compares two files. cmp will write no output if the files are the same. Under default options, if they differ, it writes to standard output the byte and line numbers at which the first difference occurred. Bytes and lines are numbered beginning with 1. If one file is an initial subsequence of the other, that fact is noted. skip1 and skip2 are initial byte offsets into file1 and file2 respectively, and may be either octal or decimal; a leading 0 denotes octal.
OPTIONS -l Write the byte number (decimal) and the differing bytes (octal) for each difference.
          -s  Write nothing for differing files; return exit statuses only.
OPERANDS The following operands are supported:
          file1  A path name of the first file to be compared. If file1 is −, the standard input will be used.
          file2  A path name of the second file to be compared. If file2 is −, the standard input will be used.
If both file1 and file2 refer to standard input or refer to the same FIFO special, block special or character special file, an error results.
USAGE See largefile(5) for the description of the behavior of cmp when encountering files greater than or equal to 2 Gbyte (2^31 bytes).
EXAMPLES EXAMPLE 1 Byte for byte comparison of files.
          The following example:
          example$ cmp file1 file2 0 1024
          does a byte for byte comparison of file1 and file2. It skips the first 1024 bytes in file2 before starting the comparison.
ENVIRONMENT VARIABLES See environ(5) for descriptions of the following environment variables that affect the execution of cmp: LC_CTYPE, LC_MESSAGES, and NLSPATH.
EXIT STATUS The following error values are returned:
          0  The files are identical.
          1  The files are different; this includes the case where one file is identical to the first part of the other.
          >1  An error occurred.
ATTRIBUTES See attributes(5) for descriptions of the following attributes:
###(cmp(1))

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

**SEE ALSO**  
comm(1), diff(1), attributes(5), environ(5), largefile(5)
col(1)

**NAME**

col – reverse line-feeds filter

**SYNOPSIS**

```bash
col [-bfpx]
```

**DESCRIPTION**

The `col` utility reads from the standard input and writes to the standard output. It performs the line overlays implied by reverse line-feeds, and by forward and reverse half-line-feeds. Unless `-x` is used, all blank characters in the input will be converted to tab characters wherever possible. `col` is particularly useful for filtering multi-column output made with the `.rt` command of `nroff(1)` and output resulting from use of the `tbl(1)` preprocessor.

The ASCII control characters SO and SI are assumed by `col` to start and end text in an alternative character set. The character set to which each input character belongs is remembered, and on output SO and SI characters are generated as appropriate to ensure that each character is written in the correct character set.

On input, the only control characters accepted are space, backspace, tab, carriage-return and newline characters, SI, SO, VT, reverse line-feed, forward half-line-feed and reverse half-line-feed. The VT character is an alternative form of full reverse line-feed, included for compatibility with some earlier programs of this type. The only other characters to be copied to the output are those that are printable.

The ASCII codes for the control functions and line-motion sequences mentioned above are as given in the table below. ESC stands for the ASCII escape character, with the octal code 033; ESC– means a sequence of two characters, ESC followed by the character x.

<table>
<thead>
<tr>
<th>Line Motion</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>reverse line-feed</td>
<td>ESC–7</td>
</tr>
<tr>
<td>reverse half-line-feed</td>
<td>ESC–8</td>
</tr>
<tr>
<td>forward half-line-feed</td>
<td>ESC–9</td>
</tr>
<tr>
<td>vertical-tab (VT)</td>
<td>013</td>
</tr>
<tr>
<td>start-of-text (SO)</td>
<td>016</td>
</tr>
<tr>
<td>end-of-text (SI)</td>
<td>017</td>
</tr>
</tbody>
</table>

**OPTIONS**

- `b` Assume that the output device in use is not capable of backspacing. In this case, if two or more characters are to appear in the same place, only the last one read will be output.

- `f` Although `col` accepts half-line motions in its input, it normally does not emit them on output. Instead, text that would appear between lines is moved to the next lower full-line boundary. This treatment can be suppressed by the `-f` (fine) option; in this case, the output from `col` may contain forward half-line-feeds (ESC–9), but will still never contain either kind of reverse line motion.
Normally, `col` will ignore any escape sequences unknown to it that are found in its input; the `-p` option may be used to cause `col` to output these sequences as regular characters, subject to overprinting from reverse line motions. The use of this option is highly discouraged unless the user is fully aware of the textual position of the escape sequences.

`-x` Prevent `col` from converting blank characters to tab characters on output wherever possible. Tab stops are considered to be at each column position \( n \) such that \( n \) modulo 8 equals 1.

**ENVIRONMENT VARIABLES**

See `environ(5)` for descriptions of the following environment variables that affect the execution of `col`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

**EXIT STATUS**

The following error values are returned:

- 0  Successful completion.
- >0  An error occurred.

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

**SEE ALSO**

`nroff(1)`, `tbl(1)`, `ascii(5)`, `attributes(5)`, `environ(5)`

**NOTES**

The input format accepted by `col` matches the output produced by `nroff` with either the `-T37` or `-Tlp` options. Use `-T37` (and the `-f` option of `col`) if the ultimate disposition of the output of `col` will be a device that can interpret half-line motions, and `-Tlp` otherwise.

`col` cannot back up more than 128 lines or handle more than 800 characters per line.

Local vertical motions that would result in backing up over the first line of the document are ignored. As a result, the first line must not have any superscripts.
comm(1)

NAME
comm – select or reject lines common to two files

SYNOPSIS
comm [-123] file1 file2

DESCRIPTION
The comm utility will read file1 and file2, which should be ordered in the current
collating sequence, and produce three text columns as output: lines only in file1; lines
only in file2; and lines in both files.

If the input files were ordered according to the collating sequence of the current locale,
the lines written will be in the collating sequence of the original lines. If not, the
results are unspecified.

OPTIONS
The following options are supported:
-1 Suppress the output column of lines unique to file1.
-2 Suppress the output column of lines unique to file2.
-3 Suppress the output column of lines duplicated in file1 and file2.

OPERANDS
The following operands are supported:
file1 A path name of the first file to be compared. If file1 is −, the standard input
is used.
file2 A path name of the second file to be compared. If file2 is −, the standard
input is used.

USAGE
See largefile(5) for the description of the behavior of comm when encountering files
greater than or equal to 2 Gbyte (231 bytes).

EXAMPLES
EXAMPLE 1 Printing a list of utilities specified by files.

If file1, file2, and file3 each contained a sorted list of utilities:
example% comm -23 file1 file2 | comm -23 - file3
would print a list of utilities in file1 not specified by either of the other files;
example% comm -12 file1 file2 | comm -12 - file3
would print a list of utilities specified by all three files; and
example% comm -12 file2 file3 | comm -23 - file1
would print a list of utilities specified by both file2 and file3, but not specified in file1.

ENVIRONMENT
VARIABLES
See environ(5) for descriptions of the following environment variables that affect the
execution of comm: LC_COLLATE, LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS
The following exit values are returned:
0 All input files were successfully output as specified.
>0 An error occurred.

ATTRIBUTES
See attributes(5) for descriptions of the following attributes:
### comm(1)

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

**SEE ALSO**

cmp(1), diff(1), sort(1), uniq(1), attributes(5), environ(5), largefile(5)
The `command` utility causes the shell to treat the arguments as a simple command, suppressing the shell function lookup.

If the `command_name` is the same as the name of one of the special built-in utilities, the special properties will not occur. In every other respect, if `command_name` is not the name of a function, the effect of `command` will be the same as omitting `command`.

The `command` utility also provides information concerning how a command name will be interpreted by the shell; see `-v` and `-V`.

The following options are supported:

- `-p` Perform the command search using a default value for `PATH` that is guaranteed to find all of the standard utilities.
- `-v` Write a string to standard output that indicates the path or command that will be used by the shell, in the current shell execution environment to invoke `command_name`.
  - Utilities, regular built-in utilities, `command_names` including a slash character, and any implementation-provided functions that are found using the `PATH` variable will be written as absolute path names.
  - Shell functions, special built-in utilities, regular built-in utilities not associated with a `PATH` search, and shell reserved words will be written as just their names.
  - An alias will be written as a command line that represents its alias definition.
  - Otherwise, no output will be written and the exit status will reflect that the name was not found.
- `-V` Write a string to standard output that indicates how the name given in the `command_name` operand will be interpreted by the shell, in the current shell execution environment. Although the format of this string is unspecified, it will indicate in which of the following categories `command_name` falls and include the information stated:
  - Utilities, regular built-in utilities, and any implementation-provided functions that are found using the `PATH` variable will be identified as such and include the absolute path name in the string.
  - Other shell functions will be identified as functions.
  - Aliases will be identified as aliases and their definitions will be included in the string.
  - Special built-in utilities will be identified as special built-in utilities.
  - Regular built-in utilities not associated with a `PATH` search will be identified as regular built-in utilities.
Shell reserved words will be identified as reserved words.

OPERANDS
The following operands are supported:

`argument` One of the strings treated as an argument to `command_name`.

`command_name` The name of a utility or a special built-in utility.

EXAMPLES

**EXAMPLE 1** Make a version of `cd` that always prints out the new working directory exactly once:

```bash
cd() {
    command cd "@" > /dev/null
    pwd
}
```

**EXAMPLE 2** Start off a “secure shell script” in which the script avoids being spoofed by its parent:

```bash
IFS=''

# The preceding value should be <space><tab><newline>.
# Set IFS to its default value.
\unalias -a

# Unset all possible aliases.
# Note that unalias is escaped to prevent an alias
# being used for unalias.
unset -f command

# Ensure command is not a user function.
PATH="$(command -p getconf _CS_PATH):$PATH"

# Put on a reliable PATH prefix.
# ...
```

At this point, given correct permissions on the directories called by `PATH`, the script has the ability to ensure that any utility it calls is the intended one. It is being very cautious because it assumes that implementation extensions may be present that would allow user functions to exist when it is invoked; this capability is not specified by this document, but it is not prohibited as an extension. For example, the `ENV` variable precedes the invocation of the script with a user startup script. Such a script could define functions to spoof the application.

ENVIRONMENT VARIABLES
See `environ(5)` for descriptions of the following environment variables that affect the execution of `command`: `LC_CTYPE`, `LC_MESSAGES`, and `NLS_PATH`.

`PATH` Determine the search path used during the command search, except as described under the `-p` option.

EXIT STATUS
When the `-v` or `-V` options are specified, the following exit values are returned:

- `0` Successful completion.
- `>0` The `command_name` could not be found or an error occurred.

Otherwise, the following exit values are returned:
The utility specified by `command_name` was found but could not be invoked.

An error occurred in the `command` utility or the utility specified by `command_name` could not be found.

Otherwise, the exit status of `command` will be that of the simple command specified by the arguments to `command`.

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>Attribute Type</th>
<th>Attribute Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

**SEE ALSO**

`sh(1), type(1), attributes(5)`
compress(1)

NAME
compress, uncompress, zcat – compress, uncompress files or display expanded files

SYNOPSIS
compress [-fv] [-b bits] [file...]
compress [-cfv] [-b bits] [file]
uncompress [-cfv] [file...]
zcat [file...]

compress
The compress utility will attempt to reduce the size of the named files by using adaptive Lempel-Ziv coding. Except when the output is to the standard output, each file will be replaced by one with the extension .Z, while keeping the same ownership modes, change times and modification times. If appending the .Z to the file pathname would make the pathname exceed 1023 bytes, the command will fail. If no files are specified, the standard input will be compressed to the standard output.

The amount of compression obtained depends on the size of the input, the number of bits per code, and the distribution of common substrings. Typically, text such as source code or English is reduced by 50–60%. Compression is generally much better than that achieved by Huffman coding (as used in pack(1)), and takes less time to compute. The bits parameter specified during compression is encoded within the compressed file, along with a magic number to ensure that neither decompression of random data nor recompression of compressed data is subsequently allowed.

uncompress
The uncompress utility will restore files to their original state after they have been compressed using the compress utility. If no files are specified, the standard input will be uncompressed to the standard output.

This utility supports the uncompressing of any files produced by compress. For files produced by compress on other systems, uncompress supports 9- to 16-bit compression (see -b).

zcat
The zcat utility will write to standard output the uncompressed form of files that have been compressed using compress. It is the equivalent of uncompress -c. Input files are not affected.

OPTIONS
The following options are supported:
-c   Writes to the standard output; no files are changed and no .Z files are created. The behavior of zcat is identical to that of 'uncompress -c'.
-f   When compressing, forces compression of file, even if it does not actually reduce the size of the file, or if the corresponding file.Z file already exists. If the -f option is not given, and the process is not running in the background, prompts to verify whether an existing file.Z file should be overwritten. When uncompressing, does not prompt for overwriting files. If the -f option is not given, and the process is not running in the background, prompts to verify whether an existing file should be overwritten. If the standard input is not a terminal and -f is not given, writes a diagnostic message to standard error and exits with a status greater than 0.
compress(1)

-\( v \)  Verbose. Writes to standard error messages concerning the percentage reduction or expansion of each file.

-\( b \) bits  Sets the upper limit (in bits) for common substring codes. bits must be between 9 and 16 (16 is the default). Lowering the number of bits will result in larger, less compressed files.

**OPERANDS**  The following operand is supported:

`file`  A path name of a file to be compressed by `compress`, uncompressed by `uncompress`, or whose uncompressed form is written to standard out by `zcat`. If `file` is `-`, or if no `file` is specified, the standard input will be used.

**USAGE**  See `largefile(5)` for the description of the behavior of `compress`, `uncompress`, and `zcat` when encountering files greater than or equal to 2 Gbyte (\( 2^{31} \) bytes).

See `environ(5)` for descriptions of the following environment variables that affect the execution of `compress`, `uncompress`, and `zcat`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

**EXIT STATUS**  The following error values are returned:

0  Successful completion.
1  An error occurred.
2  One or more files were not compressed because they would have increased in size (and the `-f` option was not specified).

>2  An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

**SEE ALSO**  `ln(1)`, `pack(1)`, `attributes(5)`, `environ(5)`, `largefile(5)`

**DIAGNOSTICS**  Usage: `compress [-fvc] [-b maxbits] [file ...]`

Invalid options were specified on the command line.

Missing maxbits

Maxbits must follow `-b`, or invalid maxbits, not a numeric value.

`file`: not in compressed format

The file specified to `uncompress` has not been compressed.
file: compressed with \textit{xx}bits, can only handle \textit{yy}bits

The file was compressed by a program that could deal with more \textit{bits} than the compress code on this machine. Recompress the file with smaller \textit{bits}.

file: already has .Z suffix -- no change

The file is assumed to be already compressed. Rename the file and try again.

file: already exists; do you wish to overwrite (\textit{y} or \textit{n})?

Respond \textit{y} if you want the output file to be replaced; \textit{n} if not.

uncompress: corrupt input

A \texttt{SIGSEGV} violation was detected, which usually means that the input file is corrupted.

Compression: \textit{xx.xx}\%

Percentage of the input saved by compression. (Relevant only for \texttt{-v}.)

-- not a regular file: unchanged

When the input file is not a regular file, (such as a directory), it is left unaltered.

-- has \textit{xx} other links: unchanged

The input file has links; it is left unchanged. See \texttt{ln(1)} for more information.

-- file unchanged

No savings are achieved by compression. The input remains uncompressed.

filename too long to tack on .Z

The path name is too long to append the .Z suffix.

NOTES

Although compressed files are compatible between machines with large memory, \texttt{-b 12} should be used for file transfer to architectures with a small process data space (64KB or less).

\texttt{compress} should be more flexible about the existence of the .Z suffix.
coproc(1F)

NAME  coproc, cocreate, cosend, cocheck, coreceive, codestroy – communicate with a process

SYNOPSIS  
cocreate [-r rpath] [-w wpath] [-i id] [-R refname] [-s send_string]
        [-e expect_string] command

cosend [-n] proc_id string

ccheck proc_id

creceive proc_id

cdestroy [-R refname] proc_id [string]

DESCRIPTION  These co-processing functions provide a flexible means of interaction between FMLI and an independent process; especially, they enable FMLI to be responsive to asynchronous activity.

The cocreate function starts command as a co-process and initializes communications by setting up pipes between FMLI and the standard input and standard output of command. The argument command must be an executable and its arguments (if any). This means that command expects strings on its input (supplied by cosend) and sends information on its output that can be handled in various ways by FMLI.

The cosend function sends string to the co-process identified by proc_id via the pipe set up by cocreate (optionally wpath), where proc_id can be either the command or id specified in cocreate. By default, cosend blocks, waiting for a response from the co-process. Also by default, FMLI does not send a send_string and does not expect an expect_string (except a newline). That is, it reads only one line of output from the co-process. If -e expect_string was not defined when the pipe was created, then the output of the co-process is any single string followed by a newline: any other lines of output remain on the pipe. If the -e option was specified when the pipe was created, cosend reads lines from the pipe until it reads a line starting with expect_string. All lines except the line starting with expect_string become the output of cosend.

The ccheck function determines if input is available from the process identified by proc_id, where proc_id can be either the command or id specified in cocreate. It returns a Boolean value, which makes ccheck useful in if statements and in other backquoted expressions in Boolean descriptors. ccheck receives no input from the co-process; it simply indicates if input is available from the co-process. You must use coreceive to actually accept the input. The ccheck function can be called from a reread descriptor to force a frame to update when new data is available. This is useful when the default value of a field in a form includes coreceive.

The coreceive function is used to read input from the co-process identified by proc_id, where proc_id can be either the command or id specified in cocreate. It should only be used when it has been determined, using ccheck, that input is actually available. If the -e option was used when the co-process was created, coreceive will continue to return lines of input until expect_string is read. At this point, coreceive will terminate. The output of coreceive is all the lines that were read excluding the
line starting with `expect_string`. If the `-e` option was not used in the `cocreate`, each
invocation of `coreceive` will return exactly one line from the co-process. If no input
is available when `coreceive` is invoked, it will simply terminate without producing
output.

The `codestroy` function terminates the read/write pipes to `proc-id`, where `proc_id` can
be either the `command or id` specified in `cocreate`. It generates a SIGPIPE signal to
the (child) co-process. This kills the co-process, unless the co-process ignores the
SIGPIPE signal. If the co-process ignores the SIGPIPE, it will not die, even after the
FMLI process terminates (the parent process id of the co-process will be 1).

The optional argument `string` is sent to the co-process before the co-process dies. If
`string` is not supplied, a NULL string is passed, followed by the normal `send_string`
(newline by default). That is, `codestroy` will call `cosend proc_id string`; this implies
that `codestroy` will write any output generated by the co-process to `stdout`. For
example, if an interactive co-process is written to expect a "quit" string when the
communication is over, the `close` descriptor could be defined; `close='codestroy
ID 'quit' | message'` and any output generated by the co-process when the
string `quit` is sent to it via `codestroy` (using `cosend`) would be redirected to the
message line.

The `codestroy` function should usually be given the `-R` option, since you may have
more than one process with the same name, and you do not want to kill the wrong
one. `codestroy` keeps track of the number of `refnames` you have assigned to a process
with `cocreate`, and when the last instance is killed, it kills the process (id) for you.
`codestroy` is typically called as part of a `close` descriptor because `close` is
evaluated when a frame is closed. This is important because the co-process will
continue to run if `codestroy` is not issued.

When writing programs to use as co-processes, the following tips may be useful. If the
co-process program is written in C language, be sure to flush output after writing to
the pipe. (Currently, `awk(1)` and `sed(1)` cannot be used in a co-process program
because they do not flush after lines of output.) Shell scripts are well-mannered, but
slow. C language is recommended. If possible, use the default `send_string, rpath` and
`wpath`. In most cases, `expect_string` will have to be specified. This, of course, depends
on the co-process.

In the case where asynchronous communication from a co-process is desired, a
co-process program should use `vsig` to force strings into the pipe and then signal
FMLI that output from the co-process is available. This causes the `reread` descriptor
of all frames to be evaluated immediately.

**OPTIONS**

`cocreate` options are:

- `-r rpath`
  If `-r` is specified, `rpath` is the pathname from which
  FMLI reads information. This option is usually used to
  set up communication with processes that naturally
  write to a certain path. If `-r` is not specified, `cocreate`
  will choose a unique path in `/var/tmp`.

*User Commands* 181
-w wpath
  If -w is specified, wpath is the pathname to which cosend writes information. This option is usually used so that one process can talk to many different FMLI processes through the same pipe. If -w is not specified, cocreate will choose a unique path in /var/tmp.

-i id
  If -i is specified, id is an alternative name for the co-process initialized by this cocreate. If -i is not specified, id defaults to command. The argument id can later be used with the other co-processing functions rather than command. This option is typically used, since it facilitates the creation of two or more co-processes generated from the same command. (For example, cocreate -i ID1 program args and cocreate -i ID2 program different_args).

-R refname
  If -R is specified, refname is a local name for the co-process. Since the cocreate function can be issued more than once, a refname is useful when the same co-process is referenced a second or subsequent time. With the -R option, if the co-process already exists a new one will not be created: the same pipes will be shared. Then, refname can be used as an argument to the -R option to codestroy when you want to end a particular connection to a co-process and leave other connections undisturbed. (The co-process is only killed after codestroy -R has been called as many times as cocreate -R was called.)

-s send_string
  The -s option specifies send_string as a string that will be appended to all output sent to the co-process using cosend. This option allows a co-process to know when input from FMLI has completed. The default send_string is a newline if -s is not specified.

-e expect_string
  The -e option specifies expect_string as a string that identifies the end of all output returned by the co-process. (Note: expect_string need only be the initial part of a line, and there must be a newline at the end of the co-process output.) This option allows FMLI to know when output from the co-process has completed. The default expect_string is a newline if -e is not specified.

cosend options are:

- n
  If the -n option is specified, cosend will not wait for a response from the co-process. It simply returns, providing no output. If the -n option is not used, a co-process that does not answer will cause FMLI to permanently hang, waiting for input from the co-process.
EXAMPLES

EXAMPLE 1 Sample commands

```
init='cocreate -i BIGPROCESS initialize'
close='codestroy BIGPROCESS'

reread='cocheck BIGPROCESS'

name='cosend -n BIGPROCESS field1'

name="Receive field"
inactive=TRUE
value='coreceive BIGPROCESS'
```

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO

awk(1), cat(1), sed(1), vsig(1F), attributes(5)

NOTES

If cosend is used without the -n option, a co-process that does not answer will cause FMLI to permanently hang.

The use of non-alphabetic characters in input and output strings to a co-process should be avoided because they may not get transferred correctly.
NAME

cp – copy files

SYNOPSIS

/usr/bin/cp [-fip@] source_file target_file
/usr/bin/cp [-fip@] source_file… target
/usr/bin/cp -r | -R [-fip@] source_dir… target
/usr/xpg4/bin/cp [-fip@] source_file target_file
/usr/xpg4/bin/cp [-fip@] source_file… target
/usr/xpg4/bin/cp -r | -R [-fip@] source_dir… target

DESCRIPTION

In the first synopsis form, neither source_file nor target_file are directory files, nor can
they have the same name. The cp utility will copy the contents of source_file to the
destination path named by target_file. If target_file exists, cp will overwrite its contents,
but the mode (and ACL if applicable), owner, and group associated with it are not
changed. The last modification time of target_file and the last access time of source_file
are set to the time the copy was made. If target_file does not exist, cp creates a new file
named target_file that has the same mode as source_file except that the sticky bit is not
set unless the user is super-user. In this case, the owner and group of target_file are
those of the user, unless the setgid bit is set on the directory containing the newly
created file. If the directory’s setgid bit is set, the newly created file will have the
group of the containing directory rather than of the creating user. If target_file is a link
to another file, cp will overwrite the link destination with the contents of source_file;
the link(s) from target_file will remain.

In the second synopsis form, one or more source_files are copied to the directory
specified by target. For each source_file specified, a new file with the same mode (and
ACL if applicable), is created in target; the owner and group are those of the user
making the copy. It is an error if any source_file is a file of type directory, if target either
does not exist or is not a directory.

In the third synopsis form, one or more directories specified by source_dir are copied to
the directory specified by target. Either -r or -R must be specified. For each source_dir,
cp will copy all files and subdirectories.

OPTIONS

The following options are supported for both /usr/bin/cp and
/usr/xpg4/bin/cp:

- f Unlink. If a file descriptor for a destination file cannot be obtained, attempt
to unlink the destination file and proceed.

- i Interactive. cp will prompt for confirmation whenever the copy would
overwrite an existing target. A y answer means that the copy should
proceed. Any other answer prevents cp from overwriting target.

- r Recursive. cp will copy the directory and all its files, including any
subdirectories and their files to target.

- R Same as -r, except pipes are replicated, not read from.
Preserves extended attributes. cp will attempt to copy all of the source file’s extended attributes along with the file data to the destination file.

The following option is supported for /usr/bin/cp only:

-tp Preserve. cp duplicates not only the contents of source file, but also preserves the owner and group id, permission modes, modification and access time, ACLs, and extended attributes, if applicable. Notice that the command may fail if ACLs are copied to a file system without appropriate support. The command will not fail if unable to preserve extended attributes, modification and access time, or permission modes. If unable to preserve owner and group id, cp will not fail, and it will clear S_ISUID and S_ISGID bits in the target. cp will print a diagnostic message to stderr and return a non-zero exit status if unable to clear these bits.

In order to preserve the owner and group id, permission modes, and modification and access times, users must have the appropriate file access permissions. This includes being superuser or the same owner id as the destination file.

The following option is supported for /usr/xpg4/bin/cp only:

-tp Preserve. cp duplicates not only the contents of source file, but also preserves the owner and group id, permission modes, modification and access time, ACLs, and extended attributes, if applicable. Notice that the command may fail if ACLs or extended attributes are copied to a file system without appropriate support. If unable to duplicate the modification and access time or the permission modes, cp will print a diagnostic message to stderr and return a non-zero exit status. If unable to preserve owner and group id, cp will not fail, and it will clear S_ISUID and S_ISGID bits in the target. cp will print a diagnostic message to stderr and return a non-zero exit status if unable to clear these bits.

In order to preserve the owner and group id, permission modes, and modification and access times, users must have the appropriate file access permissions. This includes being superuser or the same owner id as the destination file.

The following operands are supported:

- source_file A pathname of a regular file to be copied.
- source_dir A pathname of a directory to be copied.
- target_file A pathname of an existing or non-existing file, used for the output when a single file is copied.
- target A pathname of a directory to contain the copied files.

See largefile(5) for the description of the behavior of cp when encountering files greater than or equal to 2 Gbyte (2^31 bytes).
EXAMPLES

EXAMPLE 1 Copying a file
example% cp goodies goodies.old
example% ls goodies*
goodies goodies.old

EXAMPLE 2 Copying a list of files to a destination directory
example% cp ~/src/* /tmp

EXAMPLE 3 Copying a directory, first to a new, and then to an existing destination directory
example% ls ~/bkup
/usr/example/fred/bkup not found
example% cp -r ~/src ~/bkup
example% ls -R ~/bkup
x.c y.c z.sh
example% cp -r ~/src ~/bkup
example% ls -R ~/bkup
src x.c y.c z.sh
src:
x.c y.c z.s

ENVIRONMENT VARIABLES
See environ(5) for descriptions of the following environment variables that affect the execution of cp: LC_COLLATE, LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS
The following exit values are returned:
0 All files were copied successfully.
>0 An error occurred.

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

<table>
<thead>
<tr>
<th>/usr/bin/cp</th>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
<td></td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
<td></td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Stable</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>/usr/xpg4/bin/cp</th>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWxcu4</td>
<td></td>
</tr>
</tbody>
</table>
The permission modes of the source file are preserved in the copy.

A `--` permits the user to mark the end of any command line options explicitly, thus allowing `cp` to recognize filename arguments that begin with a `-`.  

SEE ALSO

chmod(1), chown(1), setfacl(1), utime(2), attributes(5), environ(5), fsattr(5), largefile(5), XPG4(5)

NOTES

The permission modes of the source file are preserved in the copy.
The `cpio` command copies files into and out of a `cpio` archive. The `cpio` archive may span multiple volumes. The `-i`, `-o`, and `-p` options select the action to be performed. The following list describes each of the actions. These actions are mutually exclusive.

**Copy In Mode**

`cpio -i` (copy in) extracts files from the standard input, which is assumed to be the product of a previous `cpio -o` command. Only files with names that match one of the `pattern` are selected. See `sh(1)` and `OPERANDS` for more information about `pattern`. Extracted files are conditionally copied into the current directory tree, based on the options described below. The permissions of the files will be those of the previous `cpio -o` command. The owner and group will be the same as the current user, unless the current user is the super-user. If this is the case, owner and group will be the same as those resulting from the previous `cpio -o` command. Notice that if `cpio -i` tries to create a file that already exists and the existing file is the same age or younger (newer), `cpio` will output a warning message and not replace the file. The `-u` option can be used to unconditionally overwrite the existing file.

**Copy Out Mode**

`cpio -o` (copy out) reads a list of file path names from the standard input and copies those files to the standard output, together with path name and status information in the form of a `cpio` archive. Output is padded to an 8192-byte boundary by default or to the user-specified block size (with the `-B` or `-C` options) or to some device-dependent block size where necessary (as with the CTC tape).

**Pass Mode**

`cpio -p` (pass) reads a list of file path names from the standard input and conditionally copies those files into the destination directory tree, based on the options described below.

Note: `cpio` assumes four-byte words.

If, when writing to a character device (-o) or reading from a character device (-i), `cpio` reaches the end of a medium (such as the end of a diskette), and the `-O` and `-I` options are not used, `cpio` prints the following message:

To continue, type device/file name when ready.

To continue, you must replace the medium and type the character special device name (`/dev/rdiskette` for example) and press RETURN. You may want to continue by directing `cpio` to use a different device. For example, if you have two floppy drives you may want to switch between them so `cpio` can proceed while you are changing the floppies. Press RETURN to cause the `cpio` process to exit.

**OPTIONS**

The following options are supported:
The following options can be appended in any sequence to the -i, -o, or -p options:

- **a** Resets access times of input files after they have been copied, making cpio's access invisible. Access times are not reset for linked files when cpio -pla is specified.

- **A** Appends files to an archive. The -A option requires the -o option. Valid only with archives that are files, or that are on floppy diskettes or hard disk partitions. The effect on files that are linked in the existing portion of the archive is unpredictable.

- **b** Reverses the order of the bytes within each word. Use only with the -i option.

- **B** Blocks input/output 5120 bytes to the record. The default buffer size is 8192 bytes when this and the -C options are not used. -B does not apply to the -p (pass) option.

- **c** Reads or writes header information in ASCII character form for portability. There are no UID or GID restrictions associated with this header format. Use this option between SVR4-based machines, or the -H odc option between unknown machines. The -c option implies the use of expanded device numbers, which are only supported on SVR4-based systems. When transferring files between SunOS 4 or Interactive UNIX and the Solaris 2.6 Operating environment or compatible versions, use -H odc.

- **C bufsize** Blocks input/output bufsize bytes to the record, where bufsize is replaced by a positive integer. The default buffer size is 8192 bytes when this and -B options are not used. -C does not apply to the -p (pass) option.

- **d** Creates directories as needed.

- **E file** Specifies an input file (file) that contains a list of filenames to be extracted from the archive (one filename per line).

- **E** Copies in all files except those in patterns. See OPERANDS for a description of pattern.
-H header

Reads or writes header information in header format. Always use this option or the -c option when the origin and the destination machines are different types. This option is mutually exclusive with options -c and -6.

Valid values for header are:

- **bar**
  - Used only with the -i option (read only).

- **crc | CRC**
  - ASCII header with expanded device numbers and an additional per-file checksum. There are no UID or GID restrictions associated with this header format.

- **odc**
  - ASCII header with small device numbers. This is the IEEE/P1003 Data Interchange Standard cpio header and format. It has the widest range of portability of any of the header formats. It is the official format for transferring files between POSIX-conforming systems (see standards(5)). Use this format to communicate with SunOS 4 and Interactive UNIX. This header format allows UIDs and GIDs up to 262143 to be stored in the header.

- **tar | TAR**
  - tar header and format. This is an older tar header format that allows UIDs and GIDs up to 2097151 to be stored in the header. It is provided for the reading of legacy archives only, that is, in conjunction with option -i.

  Specifying this archive format with option -o has the same effect as specifying the "ustar" format: the output archive is in ustar format, and must be read using -H ustar.

- **ustar | USTAR**
  - IEEE/P1003 Data Interchange Standard tar header and format. This header format allows UIDs and GIDs up to 2097151 to be stored in the header.

  Files with UIDs and GIDs greater than the limit stated above will be archived with the UID and GID of 60001. To transfer a large file (8 Gb — 1 byte), the header format can be tar | TAR, ustar | USTAR, or odc only.

- **-I file**

  Reads the contents of file as an input archive, instead of the standard input. If file is a character special device, and the current
medium has been completely read, replace the medium and press RETURN to continue to the next medium. This option is used only with the -i option.

-k Attempts to skip corrupted file headers and I/O errors that may be encountered. If you want to copy files from a medium that is corrupted or out of sequence, this option lets you read only those files with good headers. For cpio archives that contain other cpio archives, if an error is encountered, cpio may terminate prematurely. cpio will find the next good header, which may be one for a smaller archive, and terminate when the smaller archive’s trailer is encountered. Use only with the -i option.

-l In pass mode, makes hard links between the source and destination whenever possible. If the -L option is also specified, the hard link will be to the file referred to by the symbolic link. Otherwise, the hard link will be to the symbolic link itself. Use only with the -p option.

-L Follows symbolic links. If a symbolic link to a directory is encountered, archives the directory referred to by the link, using the name of the link. Otherwise, archives the file referred to by the link, using the name of the link.

-m Retains previous file modification time. This option is ineffective on directories that are being copied.

-M message Defines a message to use when switching media. When you use the -O or -I options and specify a character special device, you can use this option to define the message that is printed when you reach the end of the medium. One %d can be placed in message to print the sequence number of the next medium needed to continue.

-O file Directs the output of cpio to file, instead of the standard output. If file is a character special device and the current medium is full, replace the medium and type a carriage return to continue to the next medium. Use only with the -o option.

-P Preserves ACLs. If the option is used for output, existing ACLs are written along with other attributes, except for extended attributes, to the standard output. ACLs are created as special files with a special file type. If the option is used for input, existing ACLs are extracted along with other attributes from standard input. The option recognizes the special file type. Notice that errors will occur if a cpio archive with ACLs is extracted by previous versions of cpio. This option should not be used with the -c option, as ACL support may not be present on all systems, and hence is not portable. Use ASCII headers for portability.
Interactively renames files. If the user types a carriage return
alone, the file is skipped. If the user types a ".", the original
pathname will be retained. Not available with cpio -p.

- R id
Reassigns ownership and group information for each file to user
ID. (ID must be a valid login ID from /etc/passwd.) This option
is valid only for the super-user.

- s
Swaps bytes within each half word.

- S
Swaps halfwords within each word.

- t
Prints a table of contents of the input. If any file in the table of
contents has extended attributes, these are also listed. No files are
created. - t and - v are mutually exclusive.

- u
Copies unconditionally. Normally, an older file will not replace a
newer file with the same name.

- v
Verbose. Prints a list of file and extended attribute names. When
used with the - t option, the table of contents looks like the output
of an ls -l command (see ls(1)).

- V
Special verbose. Prints a dot for each file read or written. Useful to
assure the user that cpio is working without printing out all file
names.

- 6
Processes a UNIX System Sixth Edition archive format file. Use
only with the - i option. This option is mutually exclusive with - c
and - H.

- @
Includes extended attributes in archive. By default, cpio does not
place extended attributes in the archive. With this flag, cpio will
look for extended attributes on the files to be placed in the archive
and add them, as regular files, to the archive. The extended
attribute files go in the archive as special files with special file
types. When the - @ flag is used with - i or - p, it instructs cpio to
restore extended attribute data along with the normal file data.
Extended attribute files can only be extracted from an archive as
part of a normal file extract. Attempts to explicitly extract attribute
records are ignored.

OPERANDS
The following operands are supported:

directory
A path name of an existing directory to be used as the target of
cpio - p.

pattern
Expressions making use of a pattern-matching notation similar to
that used by the shell (see sh(1)) for filename pattern matching,
and similar to regular expressions. The following metacharacters
are defined:

*    Matches any string, including the empty string.
? Matches any single character.

[ . . . ] Matches any one of the enclosed characters. A pair of characters separated by ‘−’ matches any symbol between the pair (inclusive), as defined by the system default collating sequence. If the first character following the opening ‘[’ is a ‘!’ , the results are unspecified.

! The ! (exclamation point) means not. For example, the !abc* pattern would exclude all files that begin with abc.

In pattern, metacharacters ?, *, and [ . . . ] match the slash (/) character, and backslash (\) is an escape character. Multiple cases of pattern can be specified and if no pattern is specified, the default for pattern is * (that is, select all files).

Each pattern must be enclosed in double quotes. Otherwise, the name of a file in the current directory might be used.

**USAGE**

See `largefile(5)` for the description of the behavior of cpio when encountering files greater than or equal to 2 Gbyte (2^31 bytes).

**EXAMPLES**

The following examples show three uses of cpio.

**EXAMPLE 1** Using standard input

```
example$ ls | cpio -oc > ../newfile
```

When standard input is directed through a pipe to cpio -o, as in the example above, it groups the files so they can be directed (>) to a single file (../newfile). The -c option insures that the file will be portable to other machines (as would the -H option). Instead of `ls(1)`, you could use `find(1)`, `echo(1)`, `cat(1)`, and so on, to pipe a list of names to cpio. You could direct the output to a device instead of a file.

**EXAMPLE 2** Extracting files into directories

```
example$ cat newfile | cpio -icd "memo/a1" "memo/b*"
```

In this example, cpio -i uses the output file of cpio -o (directed through a pipe with cat), extracts those files that match the patterns (memo/a1, memo/b*), creates directories below the current directory as needed (-d option), and places the files in the appropriate directories. The -c option is used if the input file was created with a portable header. If no patterns were given, all files from newfile would be placed in the directory.

**EXAMPLE 3** Copying or linking files to another directory

```
example$ find . -depth -print | cpio -pdlmv newdir
```
In this example, `cpio -p` takes the file names piped to it and copies or links (`-l` option) those files to another directory, `newdir`. The `-d` option says to create directories as needed. The `-m` option says to retain the modification time. (It is important to use the `-depth` option of `find(1)` to generate path names for `cpio`. This eliminates problems that `cpio` could have trying to create files under read-only directories.) The destination directory, `newdir`, must exist.

Notice that when you use `cpio` in conjunction with `find`, if you use the `-L` option with `cpio`, you must use the `-follow` option with `find` and vice versa. Otherwise, there will be undesirable results.

For multi-reel archives, dismount the old volume, mount the new one, and continue to the next tape by typing the name of the next device (probably the same as the first reel). To stop, type a RETURN and `cpio` will end.

**ENVIRONMENT VARIABLES**
See `environ(5)` for descriptions of the following environment variables that affect the execution of `cpio`: `LC_COLLATE`, `LC_CTYPE`, `LC_MESSAGES`, `LC_TIME`, `TZ`, and `NLSPATH`.

- `TMPDIR` `cpio` creates its temporary file in `/var/tmp` by default. Otherwise, it uses the directory specified by `TMPDIR`.

**EXIT STATUS**
The following exit values are returned:

- 0  Successful completion.
- >0  An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Stable</td>
</tr>
</tbody>
</table>

**SEE ALSO**
`ar(1)`, `cat(1)`, `echo(1)`, `find(1)`, `ls(1)`, `setfacl(1)`, `sh(1)`, `tar(1)`, `vold(1M)`, `archives(4)`, `attributes(5)`, `environ(5)`, `fsattr(5)`, `largefile(5)`, `standards(5)`

**NOTES**
The maximum path name length allowed in a `cpio` archive is determined by the header type involved. The following table shows the proper value for each supported archive header type.
<table>
<thead>
<tr>
<th>Header type</th>
<th>Command line options</th>
<th>Maximum path name length</th>
</tr>
</thead>
<tbody>
<tr>
<td>BINNARY</td>
<td>&quot;-o&quot;</td>
<td>256</td>
</tr>
<tr>
<td>POSIX</td>
<td>&quot;-oh odc&quot;</td>
<td>256</td>
</tr>
<tr>
<td>ASCII</td>
<td>&quot;-oc&quot;</td>
<td>1023</td>
</tr>
<tr>
<td>CRC</td>
<td>&quot;-oh crc&quot;</td>
<td>1023</td>
</tr>
<tr>
<td>USTAR</td>
<td>&quot;-oh ustar&quot;</td>
<td>255</td>
</tr>
</tbody>
</table>

When the command line options "-o -H tar" are specified, the archive created is of type USTAR. This means that it is an error to read this same archive using the command line options "-i -H tar". The archive should be read using the command line options "-i -H ustar". The options "-i -H tar" refer to an older tar archive format.

An error message is output for files whose UID or GID are too large to fit in the selected header format. Use -H crc or -c to create archives that allow all UID or GID values.

Only the super-user can copy special files.

Blocks are reported in 512-byte quantities.

If a file has 000 permissions, contains more than 0 characters of data, and the user is not root, the file will not be saved or restored.

The inode number stored in the header (/usr/include/archives.h) is an unsigned short, which is 2 bytes. This limits the range of inode numbers from 0 to 65535. Files which are hard linked must fall in this inode range. This could be a problem when moving cpio archives between different vendors' machines.

When the Volume Management daemon is running, accesses to floppy devices through the conventional device names (for example, /dev/rdiskette) may not succeed. See void(1M) for further details.

You must use the same blocking factor when you retrieve or copy files from the tape to the hard disk as you did when you copied files from the hard disk to the tape. Therefore, you must specify the -B or -C option.

During -p and -o processing, cpio buffers the file list presented on stdin in a temporary file.
cpp(1)

NAME
cpp – the C language preprocessor

SYNOPSIS
/usr/lib/cpp [-BCHmpPRT] [-undef] [-Dname] [-Dname = def]
[-Idirectory] [-Uname] [-Ydirectory] [input-file [output-file]]

DESCRIPTION
cpp is the C language preprocessor. It is invoked as the first pass of any C compilation started with the `cc(1B)` command; however, cpp can also be used as a first-pass preprocessor for other Sun compilers.

Although cpp can be used as a macro processor, this is not normally recommended, as its output is geared toward that which would be acceptable as input to a compiler’s second pass. Thus, the preferred way to invoke cpp is through the `cc(1B)` command, or some other compilation command. For general-purpose macro-processing, see `m4(1)`.

cpp optionally accepts two filenames as arguments. input-file and output-file are, respectively, the input and output files for the preprocessor. They default to the standard input and the standard output.

OPTIONS
The following options are supported:

- **-B**
  Support the C++ comment indicator `//`. With this indicator everything on the line after the `//` is treated as a comment.

- **-C**
  Pass all comments (except those that appear on cpp directive lines) through the preprocessor. By default, cpp strips out C-style comments.

- **-H**
  Print the pathnames of included files, one per line on the standard error.

- **-M**
  Generate a list of makefile dependencies and write them to the standard output. This list indicates that the object file which would be generated from the input file depends on the input file as well as the include files referenced.

- **-p**
  Use only the first eight characters to distinguish preprocessor symbols, and issue a warning if extra tokens appear at the end of a line containing a directive.

- **-P**
  Preprocess the input without producing the line control information used by the next pass of the C compiler.

- **-R**
  Allow recursive macros.

- **-T**
  Use only the first eight characters for distinguishing different preprocessor names. This option is included for backward compatibility with systems which always use only the first eight characters.

- **-undef**
  Remove initial definitions for all predefined symbols.
cpp(1)

-\Dname
Define name as 1 (one). This is the same as if a \Dname=1 option appeared on the cpp command line, or as if a

#define name 1

line appeared in the source file that cpp is processing.

-\Dname-def
Define name as if by a \def line appeared in the source file that cpp is processing. The \D option has lower precedence than the \U option. That is, if the same name is used in both a \U option and a \D option, the name will be undefined regardless of the order of the options.

-\Idirectory
Insert directory into the search path for #include files with names not beginning with ‘/’. directory is inserted ahead of the standard list of “include” directories. Thus, #include files with names enclosed in double-quotes (”) are searched for first in the directory of the file with the #include line, then in directories named with -I options, and lastly, in directories from the standard list. For #include files with names enclosed in angle-brackets (< >), the directory of the file with the #include line is not searched. See Details below for exact details of this search order.

-\Uname
Remove any initial definition of name, where name is a symbol that is predefined by a particular preprocessor. Here is a partial list of symbols that may be predefined, depending upon the architecture of the system:

Operating System: ibm, gcos, os, tss and unix
Hardware: interdata, pdp11, u370, u3b, u3b2, u3b5, u3b15, u3b20d, vax, ns32000, iAPX286, i386, sparc, and sun
UNIX system variant: RES, and RT
The lint command: lint

The symbols sun, sparc and unix are defined for all Sun systems.

-\Ydirectory
Use directory directory in place of the standard list of directories when searching for #include files.

Directives
All cpp directives start with a hash symbol (#) as the first character on a line. White space (SPACE or TAB characters) can appear after the initial # for proper indentation.
#define name token-string
Replace subsequent instances of name with token-string.

#define name (argument [, argument] ...) token-string
There can be no space between name and the ‘(’. Replace subsequent instances of name, followed by a parenthesized list of arguments, with token-string, where each occurrence of an argument in the token-string is replaced by the corresponding token in the comma-separated list. When a macro with arguments is expanded, the arguments are placed into the expanded token-string unchanged. After the entire token-string has been expanded, cpp re-starts its scan for names to expand at the beginning of the newly created token-string.

#undef name
Remove any definition for the symbol name. No additional tokens are permitted on the directive line after name.

#include "filename"
#include <filename>
Read in the contents of filename at this location. This data is processed by cpp as if it were part of the current file. When the <filename> notation is used, filename is only searched for in the standard ‘include’ directories. See the -I and -Y options above for more detail. No additional tokens are permitted on the directive line after the final ‘‘ or ‘>’.

#line integer-constant "filename"
Generate line control information for the next pass of the C compiler. integer-constant is interpreted as the line number of the next line and filename is interpreted as the file from where it comes. If “filename” is not given, the current filename is unchanged. No additional tokens are permitted on the directive line after the optional filename.

#if constant-expression
Subsequent lines up to the matching #else, #elif, or #endif directive, appear in the output only if constant-expression yields a nonzero value. All binary non-assignment C operators, including ‘&&’, ‘|’, and ‘,’ are legal in constant-expression. The ‘?:’ operator, and the unary ‘~’, ‘!’, and ‘-’ operators, are also legal in constant-expression.

The precedence of these operators is the same as that for C. In addition, the unary operator defined, can be used in constant-expression in these two forms: ‘defined ( name )’ or ‘defined name’. This allows the effect of #ifdef and #ifndef directives (described below) in the #if directive. Only these operators, integer constants, and names that are known by cpp should be used within constant-expression. In particular, the size of operator is not available.

#ifdef name
Subsequent lines up to the matching #else, #elif, or #endif appear in the output only if name has been defined, either with a #define directive or a -D option, and in the absence of an intervening #undef directive. Additional tokens after name on the directive line will be silently ignored.
#ifndef name
Subsequent lines up to the matching #else, #elif, or #endif appear in the output only if name has not been defined, or if its definition has been removed with an #undef directive. No additional tokens are permitted on the directive line after name.

#elif constant-expression
Any number of #elif directives may appear between an #if, #ifdef, or ifndef directive and a matching #else or #endif directive. The lines following the #elif directive appear in the output only if all of the following conditions hold:

- The constant-expression in the preceding #if directive evaluated to zero, the name in the preceding #ifdef is not defined, or the name in the preceding ifndef directive was defined.
- The constant-expression in all intervening #elif directives evaluated to zero.
- The current constant-expression evaluates to non-zero.

If the constant-expression evaluates to non-zero, subsequent #elif and #else directives are ignored up to the matching #endif. Any constant-expression allowed in an #if directive is allowed in an #elif directive.

#else
This inverts the sense of the conditional directive otherwise in effect. If the preceding conditional would indicate that lines are to be included, then lines between the #else and the matching #endif are ignored. If the preceding conditional indicates that lines would be ignored, subsequent lines are included in the output. Conditional directives and corresponding #else directives can be nested.

#endif
End a section of lines begun by one of the conditional directives #if, #ifdef, or ifndef. Each such directive must have a matching #endif.

Macro definitions are recognized in define directive bodies, even when they occur inside character constants and quoted strings. For instance, the output from:

```
#define abc(a) |
```

is:

```
  #1 "x"
  | `|xyz |`
```

The second line is a NEWLINE. The last seven characters are “| `|xyz |” (vertical-bar, backquote, vertical-bar, x, y, z, vertical-bar). Macro names are not recognized within character constants or quoted strings during the regular scan. Thus:  

User Commands 199
cpp(1)

```c
#define abc xyz
printf("abc");
```

does not expand abc in the second line, since it is inside a quoted string that is not part of a `#define` macro definition.

Macros are not expanded while processing a `#define` or `#undef`. Thus:

```c
#define abc zingo
#define xyz abc
#undef abc
xyz
```

produces abc. The token appearing immediately after an `#ifdef` or `#ifndef` is not expanded.

Macros are not expanded during the scan which determines the actual parameters to another macro call. Thus:

```c
#define reverse(first,second)second first
#define greeting hello
reverse(greeting,
#define greeting goodbye
)
```

produces " `#define hello goodbye hello"."

**Output** Output consists of a copy of the input file, with modifications, plus lines of the form:

```
#lineno " filename " "level "
```

indicating the original source line number and filename of the following output line and whether this is the first such line after an include file has been entered (level=1), the first such line after an include file has been exited (level=2), or any other such line (level is empty).

**Directory Search Order** #include files are searched for in the following order:

1. The directory of the file that contains the #include request (that is, #include is relative to the file being scanned when the request is made).
2. The directories specified by `-I` options, in left-to-right order.
3. The standard directory(s) (/usr/include on UNIX systems).

**Special Names** Two special names are understood by cpp. The name `__LINE__` is defined as the current line number (a decimal integer) as known by cpp, and `__FILE__` is defined as the current filename (a C string) as known by cpp. They can be used anywhere (including in macros) just as any other defined name.
Newline Characters
A NEWLINE character terminates a character constant or quoted string. An escaped
NEWLINE (that is, a backslash immediately followed by a NEWLINE) may be used in
the body of a #define statement to continue the definition onto the next line. The
escaped NEWLINE is not included in the macro value.

Comments
Comments are removed (unless the -C option is used on the command line).
Comments are also ignored, except that a comment terminates a token.

EXIT STATUS
The following exit values are returned:
0 Successful completion.
non-zero An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsprot</td>
</tr>
</tbody>
</table>

SEE ALSO cc(1B), m4(1), attributes(5)

DIAGNOSTICS
The error messages produced by cpp are intended to be self-explanatory. The line
number and filename where the error occurred are printed along with the diagnostic.

NOTES
When NEWLINE characters were found in argument lists for macros to be expanded,
some previous versions of cpp put out the NEWLINE characters as they were found
and expanded. The current version of cpp replaces them with SPACE characters.

Because the standard directory for included files may be different in different
environments, this form of #include directive:

#include <file.h>

should be used, rather than one with an absolute path, like:

#include "/usr/include/file.h"

cpp warns about the use of the absolute pathname.

While the compiler allows 8-bit strings and comments, 8-bits are not allowed
anywhere else.
cputrack(1)

**NAME**
cputrack – monitor process and LWP behavior using CPU performance counters

**SYNOPSIS**

cputrack -c eventspec [-c eventspec] ... -p pid [-efntvD] [-N count]

[-o pathname] [-T interval]

**DESCRIPTION**
The cputrack utility allows CPU performance counters to be used to monitor the behavior of a process or family of processes running on the system. If *interval* is specified with the -T option, cputrack samples activity every *interval* seconds, repeating forever. If a *count* is specified with the -N option, the statistics are repeated *count* times for each process tracked. If neither are specified, an interval of one second is used. If command and optional args are specified, cputrack runs the command with the arguments given while monitoring the specified CPU performance events. Alternatively, the process ID of an existing process can be specified using the -p option.

Because cputrack is an unprivileged program, it is subject to the same restrictions that apply to truss(1). For example, setuid(2) executables cannot be tracked.

**OPTIONS**
The following options are supported:

- **-c eventspec** Specifies a set of events for the CPU performance counters to monitor. The list of available events and the syntax of the event specifications for the system can be determined using the -h option. The semantics of these event specifications can be determined by reading the CPU manufacturers documentation for the events. See cpc_strtoevent(3CPC) for a description of the syntax.

Multiple -c options may be specified, in which case cputrack cycles between the different event settings on each sample.

- **-D** Enables debug mode.

- **-e** Follows all exec(2), or execve(2) system calls. Without this option, cputrack terminates when the process image is overlaid with a new executable.

- **-f** Follows all children created by fork(2), fork1(2), or vfork(2) system calls.

- **-h** Prints an extended help message on how to use the utility and how to program the processor-dependent counters.

- **-n** Omits all header output (useful if cputrack is the beginning of a pipeline).

- **-N count** Specifies the maximum number of CPU performance counter samples to take before exiting.
-o outfile  Specifies file to be used for the cputrack output.
-p pid    Interprets the argument as the process ID of an existing process to which process counter context should be attached and monitored.
-t    Prints an additional column of processor cycle counts, if available on the current architecture.
-T interval  Specifies the interval between CPU performance counter samples in seconds.
-v    Enables more verbose output.

**USAGE**

The operating system enforces certain restrictions on the tracing of processes. In particular, a command whose object file cannot be read by a user cannot be tracked by that user; set-uid and set-gid commands can only be tracked by a privileged user. Unless it is run by a privileged user, cputrack loses control of any process that performs an `exec()` of a set-id or unreadable object file. Such processes continue normally, though independently of cputrack, from the point of the `exec()`.

The system may run out of per-user process slots when the `-f` option is used, since cputrack runs one controlling process for each process being tracked.

The times printed by cputrack correspond to the wallclock time when the hardware counters were actually sampled, instead of when the program told the kernel to sample them. The time is derived from the same timebase as `gethrtime(3C)`.

The cputrack utility attaches performance counter context to each process that it examines. The presence of this context allows the performance counters to be multiplexed between different processes on the system, but it cannot be used at the same time as the `cpustat(1M)` utility.

Once an instance of the `cpustat` utility is running, further attempts to run cputrack will fail until all instances of `cpustat` terminate.

Sometimes cputrack provides sufficient flexibility and prints sufficient statistics to make adding the event selection code to an application unnecessary. However, more control is occasionally desired. Because the same performance counter context is used by both the application itself and by the agent LWP injected into the application by cputrack, it is possible for an application to interact with the counter context to achieve some interesting capabilities. See `cpc_count_usr_events(3CPC)`.

The processor cycle counts enabled by the `-t` option always apply to both user and system modes, regardless of the settings applied to the performance counter registers.

The output of cputrack is designed to be readily parseable by `nawk(1)` and `perl(1)`, thereby allowing performance tools to be composed by embedding cputrack in scripts. Alternatively, tools may be constructed directly using the same APIs that cputrack is built upon, using the facilities of `libcpc(3LIB)` and `libpctx(3LIB)`. See `cpc(3CPC)`.
Although `cputrack` uses performance counter context to maintain separate performance counter values for each LWP, some of the events that can be counted will inevitably be impacted by other activities occurring on the system, particularly for limited resources that are shared between processes (for example, cache miss rates). For such events, it may also be interesting to observe overall system behavior with `cpustat(1M)`.

For the `-T interval` option, if `interval` is specified as zero, no periodic sampling is performed. The performance counters are only sampled when the process creates or destroys an LWP, or it invokes `fork(2)`, `exec(2)`, or `exit(2)`.

**EXAMPLE 1** Using performance counters to count clock cycles

In this example, the utility is being used on a machine containing an UltraSPARC 1 processor. The counters are set to count processor clock cycles and instructions dispatched in user mode while running the `sleep(1)` command.

```bash
example% cputrack -c pic0=Cycle_cnt,pic1=Instr_cnt sleep 10
time lwp event pic0 pic1
2.040 1 tick 377820 202593
4.028 1 tick 0 0
6.028 1 tick 0 0
8.028 1 tick 0 0
10.028 1 tick 6930 954
10.036 1 exit 410623 212137
```

**EXAMPLE 2** Counting external cache references and hits

This example shows more verbose output while following the `fork()` and `exec()` of a simple shell script on an UltraSPARC machine. The counters are measuring the number of external cache references and external cache hits. Notice that the explicit `pic0` and `pic1` names can be omitted where there are no ambiguities.

```bash
example% cputrack -fev -c EC_ref,EC_hit /bin/ulimit -c
```

**EXAMPLE 3** Counting instructions

This example shows how many instructions were executed in the application and in the kernel to print the date on a Pentium machine:
EXAMPLE 3 Counting instructions  (Continued)

```
example$ cputrack -c inst_retired,inst_retired,nouser1,sys1 date
    time  lwp  event   pic0  pic1
Fri Aug 20 20:03:08 PDT 1999
    0.072 1  exit   246725 339666
```

WARNINGS

By running any instance of the `cpustat(1M)` utility, all existing performance counter context is forcibly invalidated across the machine. This may in turn cause all invocations of the `cputrack` command to exit prematurely with unspecified errors.

If `cputrack` is invoked on a system that has CPU performance counters, but on which the packages containing the kernel support for those counters is not installed, the following message appears:

```
cputrack: CPU performance counters are inaccessible on this machine
```

This error message implies that `cpc_access()` has failed and is documented in `cpc_access(3CPC)`. Review this documentation for more information about the problem and possible solutions.

ATTRIBUTES

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

```
<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcpcu (32-bit)</td>
</tr>
<tr>
<td></td>
<td>SUNWcpcux (64-bit)</td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Evolving</td>
</tr>
</tbody>
</table>
```

SEE ALSO

nawk(1), perl(1), proc(1), truss(1), prstat(1M), cpustat(1M), exec(2), exit(2), fork(2), setuid(2), vfork(2), gethrtime(3C), cpc(3CPC), cpc_access(3CPC), cpc_count usr_events(3CPC), cpc_strtoevent(3CPC), libcpc(3LIB), libpctx(3LIB), proc(4), attributes(5)


NAME  crle – configure runtime linking environment

           [-I name] [-g name] [-G name] [-l dir] [-o dir] [-s dir] [-t [ ELF
           | AOUT]] [-u] [-v]

DESCRIPTION  The crle utility provides for the creation and display of a runtime linking
configuration file. Without any arguments, or with just the -c option, crle displays
the contents of a configuration file, any system defaults and the command-line
required to regenerate the configuration file. When used with any other options, a new
configuration file is created or updated. The configuration file is read and interpreted
by the runtime linker, ld.so.1(1), during process start-up.

The default configuration file is /var/ld/ld.config for 32-bit objects and
/var/ld/64/ld.config for 64-bit objects. Note: It is recommended that any new
configuration file is first created in a temporary location. The environment variable
LD_CONFIG can be set to this new configuration file to cause its use by the runtime
linker instead of any default. After verification, the new configuration file can be
moved to the default location if desired. Setting the environment variable
LD_NOCONFIG to any value results in the runtime linker ignoring any configuration
files, and may prove useful during experimentation.

The configuration file may contain the following information:

Default Search Paths  The runtime linker uses a prescribed search path for
locating the dynamic dependencies of an object. This
search path starts with the components of any
LD_LIBRARY_PATH definition, followed by the
components of an object’s runpath and finally any
defaults specific to the object’s type. This last
component of the search path can be expressed within
the configuration file. Note: Typical use of this facility
should augment any system defaults; see the -l
option.

Trusted Directories  When processing a secure application the runtime
linker restricts the use of LD_LIBRARY_PATH and the
directories from which preload and audit libraries may
be used to known trusted directories. These trusted
directories can be expressed within the configuration
file. Note: Typical use of this facility should augment
any system defaults; see the -s option.

Directory Cache  The location of shared objects within defined
directories can be maintained as a cache within the
configuration file. This directory cache can reduce the
overhead of searching for application dependencies.

Alternative Objects  In conjunction with the directory cache, shared objects
may have alternative objects specified for use at
runtime. These alternate objects may be supplied by
the user, or can be created by crle as copies of shared
objects fixed to known memory locations. These fixed
alternative objects can require less processing at
runtime than their original shared object counterpart.

Environment Variables

Any environment variable interpreted by the runtime
linker can be specified within the configuration file.

Defining alternative default search paths, or additional trusted directories can be
useful for administrators who wish to install third party software in a central location,
or otherwise alter the search path of applications that may not have been coded with
suitable runpaths.

Defining user supplied alternative objects provides a means of replacing dependencies
other than via symbolic links or requiring LD_LIBRARY_PATH settings.

Defining runtime linker environment variables provides a means of centralizing their
definition for all applications.

The directory cache and crle generated alternate objects can provide a means of
reducing the runtime start-up overhead of applications that require many
dependencies, or whose dependencies are expensive to relocate (this may be the case
when shared objects contain position-dependent code).

When alternate objects generated by crle are specified within a configuration file,
ld.so.1(1) performs some minimal consistency verification of the alternative objects
against their originating objects. This verification is intended to avert application
failure should an applications configuration information become out-of-sync with the
underlying system components. When this situation arises the flexibility offered by
dynamic linking system components may be compromised, and diagnosing the
application failure may be difficult. Note: No verification of directory cache
information is performed. Any changes to the directory structure will not be seen by a
process until the cache is rebuilt.

System shared objects are often well tuned and may have no benefit being cached. The
directory cache and alternative object features are typically applicable to user
applications and shared objects.

crle creates alternate objects for the shared objects discovered when using the -I and
-G options by calls to dldump(3DL). The alternate object is created in the directory
specified by the preceding -o option, or defaults to the directory in which the
configuration file is created. The flags used for the dldump() are specified using the
-f option, or default to RTLD_RELATIVE.

OPTIONS

The following options are supported:

-64 Specifies to process 64-bit objects, the default is 32-bit.
-a name
This option adds an alternative to name to the configuration file. The actual alternative file must be supplied by the user. Multiple occurrences of this option are permitted. If name is a directory each shared object within the directory is added to the cache. If name does not exist, it is marked in the cache as a nonexistent file.

-c conf
Specifies to use the configuration file name conf. If this option is not supplied the default configuration file is used.

-e env
This option specifies a replaceable environment variable, env. Only environment variables applicable to the runtime linker are meaningful. Multiple occurrences of this option are permitted. This option is similar to the -E option, but differs in how configuration file definitions and process environment definitions of the same name are resolved at runtime.

A definition established in a configuration file can be overridden by a process environment definition, or be suppressed by a null-value process environment definition.

In other words, these configuration file definitions can be replaced or removed by the process environment at runtime.

-E env
This option specifies a permanent environment variable, env. Only environment variables applicable to the runtime linker are meaningful. Multiple occurrences of this option are permitted. This option is similar to the -e option, but differs in how configuration file definitions and process environment definitions of the same name are resolved at runtime.

Environment variable definitions meaningful to the runtime linker fall into one of two categories, that is, singular definitions such as LD_NOLAZYLOAD=1 and LD_DEBUG_OUTPUT=file, or list definitions which can take one or more values such as LD_LIBRARY_PATH=path, and LD_DEBUG=files,details.

A singular definition established in a configuration file will take precedence over a process environment definition. A list definition established in a configuration file will be appended to a process
environment definition. Any definition established in a configuration file cannot be suppressed by a null-value process environment definition.

In other words, these configuration file definitions cannot be replaced or removed by the process environment at runtime.

- **f flags**
  This option provides the symbolic flags argument to the `dldump`(3DL) calls used to generate alternate objects. Any of the RTLD_REL flags defined in `/usr/include/dlfcn.h` can be used. Multiple flags can be or'ed together using the "|" character, and in this case the string should be quoted to avoid expansion by the shell. If no flags values are provided the default flag is `RTLD_REL_RELATIVE`.

- **i name**
  This option adds an individual name to the configuration cache. Multiple occurrences of this option are permitted. name may be a shared object or a directory. If name is a directory each shared object within the directory is added to the cache. Note: If name does not exist, it is marked in the cache as a nonexistent directory.

- **I name**
  This option is the same as -i and in addition any shared objects have alternatives created via `dldump`(3DL). If the -f flag contains `RTLD_REL_EXEC` then name may be a dynamic executable, for which an alternative is created. Only one dynamic executable can be specified in this manner as the cache created is specific to this application.

- **g name**
  This option adds the group name to the configuration cache. Each object is expanded to determine its dependencies. Multiple occurrences of this option are permitted. name may be a dynamic executable, shared object or a directory. The name itself, if it is a shared object, and its dependencies are added to the cache. If name is a directory each shared object within the directory, and its dependencies, are added to the cache.

- **G name**
  This option is the same as -g and in addition any shared objects have alternatives created via `dldump`(3DL). If name is a dynamic executable, and the -f flag contains `RTLD_REL_EXEC`, then an alternative for the dynamic executable is also created. Only one dynamic executable can be specified in this manner as the cache created is specific to this application.
This option specifies a new default search directory dir for ELF or AOUT objects. Multiple occurrences of this option are permitted. The type of object applicable to the search is specified by the preceding -t option, or defaults to ELF.

The system default search path for ELF objects is /usr/lib for 32-bit objects, and /usr/lib/64 for 64-bit objects. The system default search paths for AOUT objects is /usr/4lib, /usr/lib and /usr/local/lib.

Use of this option replaces the system default search path, and thus it is normally required that a -l option be used to specify the original system default in relation to any new paths being applied. However, if the -u option is in effect, and a configuration file does not exist, the system defaults are added to the new configuration file before the new paths specified with the -l option.

This option specifies the directory dir in which any alternate objects must exist (in the case of using the -a option), or will be created (in the case of alternatives created by crle). Without this option, alternate objects will exist in the directory in which the configuration file is created. Multiple occurrences of this option are permitted, the directory dir being used to locate alternatives for any following command-line options. Alternative objects are not permitted to override their associated originals.

This option specifies a new trusted directory dir for secure ELF or AOUT objects. See SECURITY in ld.so.1(1) for a definition of secure objects.

Multiple occurrences of this option are permitted. The type of object applicable to the search is specified by the preceding -t option, or defaults to ELF.

The system default trusted directory for secure ELF objects is /usr/lib/security for 32-bit objects and /usr/lib/security/64 for 64-bit objects. The system default trusted directories for secure AOUT objects are /usr/4lib, /usr/lib, /usr/uclib, and /usr/local/lib.
Use of this option replaces the system default trusted directories, and thus it is normally required that a `-s` option be used to specify the original system default in relation to any new directories being applied. However, if the `-u` option is in effect, and a configuration file does not exist, the system defaults are added to the new configuration file before the new directories specified with the `-s` option.

`-t ELF | AOUT`

This option toggles the object type applicable to any `-l` or `-s` options that follow. The default object type is ELF.

`-u`

This option requests that a configuration file be updated, possibly with the addition of new information. Without other options any existing configuration file is inspected and its contents recomputed. Additional arguments allow information to be appended to the recomputed contents. See NOTES.

If a configuration file does not exist it will be created as directed by the other arguments. In the case of the `-l` and `-s` options any system defaults will first be applied to the configuration file before the directories specified with these options.

`-v`

Verbose mode. When creating a configuration file, a trace of the files being processed is written to the standard out. When printing the contents of a configuration file, more extensive directory and file information is provided.

By default the runtime linker attempts to read the configuration file `/var/ld/ld.config` for each 32-bit application it processes or `/var/ld/64/ld.config` for each 64-bit application. When processing an alternative application, the runtime linker will use a `$ORIGIN/ld.config.app-name` configuration file if present (see NOTES). Applications may reference an alternative configuration file either by setting the `LD_CONFIG` environment variable (see `1d.so.1(1)`), or by recording a configuration file name in the application at the time it is built using the link-editors `-c` option (see `1d(1)`).

EXAMPLES

**EXAMPLE 1** Update (and display) of a new default search path for ELF objects

```
example% crie -u -l /local/lib
example% crie
```

Configuration file [3]: /var/ld/ld.config
Default Library Path (ELF): /usr/lib:/local/lib
Trusted Directories (ELF): /usr/lib/secure (system default)
EXAMPLE 1 Update (and display) of a new default search path for ELF objects

(Continued)

Command line:
   crle -l /usr/lib:/local/lib
example% crle -u -l /usr/local/lib
example% crle

Configuration file [3]: /var/ld/ld.config
Default Library Path (ELF): /usr/lib:/local/lib:/usr/local/lib
Trusted Directories (ELF): /usr/lib/secure (system default)

Command line:
   crle -l /usr/lib:/local/lib:/usr/local/lib

In this example, the default configuration file initially did not exist, and thus the new search path /local/lib is appended to the system default. The next update appends the search path /usr/local/lib to those already established in the configuration file.

EXAMPLE 2 Creation (and display) of a new default search path and new trusted directory for ELF objects

example% crle -l /local/lib -l /usr/lib -s /local/lib
example% crle

Configuration file [2]: /var/ld/ld.config
Default Library Path (ELF): /local/lib:/usr/lib
Trusted Directories (ELF): /local/lib

Command line:
   crle -l /local/lib:/usr/lib -s /local/lib

With this configuration, third party applications may be installed in /local/bin and their associated dependencies in /local/lib. The default search path allows the applications to locate their dependencies without the need to set LD_LIBRARY_PATH.

Note: The system default trusted directory has been replaced with this example.

EXAMPLE 3 Creation of a directory cache for ELF objects

example% crle -i /usr/dt/lib -i /usr/openwin/lib -i /usr/lib -c config
example% ldd -s ./main
....
   find object=libc.so.1; required by ./main
    search path=/usr/dt/lib:/usr/openwin/lib (RPATH ./main)
    trying path=/usr/dt/lib/libc.so.1
    trying path=/usr/openwin/lib/libc.so.1
    search path=/usr/lib (default)
    trying path=/usr/lib/libc.so.1
    libc.so.1 => /usr/lib/libc.so.1
example% LD_CONFIG=config ldd -s ./main

212  man pages section 1: User Commands • Last Revised 10 Oct 2001
EXAMPLE 3 Creation of a directory cache for ELF objects  (Continued)

....
find object=libc.so.1; required by ./main
search path=/usr/dt/lib:/usr/openwin/lib (RPATH ./main)
search path=/usr/lib (default)
trying path=/usr/lib/libc.so.1
libc.so.1 => /usr/lib/libc.so.1

With this configuration, the cache reflects that the system library libc.so.1 does not exist in the directories /usr/dt/lib or /usr/openwin/lib. Therefore, the search for this system file ignores these directories even though the application’s runpath indicates they should be searched.

EXAMPLE 4 Creation of an alternative object cache for an ELF executable

```bash
example% crle -c /local/$HOST/.xterm/ld.config.xterm \
   -f RTLD_REL_ALL -G /usr/openwin/bin/xterm
example% ln -s /local/$HOST/.xterm/xterm /local/$HOST/xterm
example% ldd /usr/local/$HOST/xterm
libXaw.so.5 => /local/$HOST/.xterm/libXaw.so.5 (alternate)
libXmu.so.4 => /local/$HOST/.xterm/libXmu.so.4 (alternate)
....
libc.so.1 => /local/$HOST/.xterm/libc.so.1 (alternate)
....
```

With this configuration, a new xterm and its dependencies are created. These new objects are fully relocated to themselves and result in faster start-up than the originating objects. Note: The execution of this application uses its own specific configuration file. This model is generally more flexible than using the environment variable LD_CONFIG, as the configuration file will not be erroneously used by other applications such as ldd(1) or truss(1).

EXAMPLE 5 Creating an alternative object cache to replace an ELF shared object

```bash
example% ldd /usr/bin/vi
   libcurses.so.1 => /usr/lib/libcurses.so.1
....
example% crle -a /usr/lib/libcurses.so.1 -o /usr/ucblib
example% crle
```

Configuration file [3]: /var/ld/ld.config
 Default Library Path (ELF): /usr/lib (system default)
 Trusted Directories (ELF): /usr/lib/secure (system default)

Directory: /usr/lib
 libcurses.so.1 (alternate: /usr/ucblib/libcurses.so.1)
....

```bash
example% ldd /usr/bin/vi
   libcurses.so.1 => /usr/ucblib/libcurses.so.1 (alternate)
....
```
EXAMPLE 5 Creating an alternative object cache to replace an ELF shared object (Continued)

With this configuration, any dependency that would normally resolve to /usr/lib/libcurses.so.1 will instead resolve to /usr/ucblib/libcurses.so.1.

EXAMPLE 6 Setting replaceable and permanent environment variables

```bash
example% crle -e LD_LIBRARY_PATH=/local/lib -E LD_PRELOAD=preload.so.1
example% crle
.....
Environment Variables:
  LD_LIBRARY_PATH=/local/lib (replaceable)
  LD_PRELOAD=preload.so.1 (permanent)
.....
example% LD_DEBUG=files LD_PRELOAD=preload.so.2 ./main
.....
18764: file=preload.so.2; preloaded
18764: file=/local/lib/preload.so.2 [ ELF ]; generating link map
.....
18764: file=preload.so.1; preloaded
18764: file=/local/lib/preload.so.1 [ ELF ]; generating link map
.....
```

With this configuration file, a replaceable search path has been specified together with a permanent preload object which becomes appended to the process environment definition.

EXIT STATUS

The creation or display of a configuration file results in a 0 being returned; otherwise any error condition is accompanied with a diagnostic message and a non-zero value being returned.

NOTES

Tagging an alternative application to use an application specific configuration file can only be achieved if the original application contains one of the `.dynamic` tags `DT_FLAGS_1` or `DT_FEATURE_1`. Without these entries any application specific configuration file must be specified using the `LD_CONFIG` environment variable. Care should be exercised with this latter method as this environment variable will be visible to any forked applications.

The use of the `-u` option requires at least version 2 of `crle`. This version level is evident from displaying the contents of a configuration file:

```bash
example% crle

Configuration file [2]: /var/ld/ld.config
.....
```
With a version 2 configuration file, `crle` is capable of constructing the command-line arguments required to regenerate the configuration file and to provide full update capabilities. Although the update of a version 1 configuration file is possible, the contents of the configuration file may be insufficient for `crle` to compute the entire update requirements.

### FILES

- `/var/ld/ld.config` Default configuration file for 32-bit applications.
- `/var/ld/64/ld.config` Default configuration file for 64-bit applications.
- `/var/tmp` Default location for temporary configuration file (see `tempnam(3C)`).
- `/usr/lib/lddstub` Stub application employed to `dldump(3DL)` 32-bit objects.
- `/usr/lib/64/lddstub` Stub application employed to `dldump(3DL)` 64-bit objects.
- `/usr/lib/libcrle.so.1` Audit library employed to `dldump(3DL)` 32-bit objects.
- `/usr/lib/64/libcrle.so.1` Audit library employed to `dldump(3DL)` 64-bit objects.

### ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWtoo</td>
</tr>
</tbody>
</table>

### SEE ALSO

`ld(1), ld.so.1(1), dldump(3DL), tempnam(3C), attributes(5)`
### NAME
`crontab` – user crontab file

### SYNOPSIS
```
`crontab` [filename]
`crontab` [-elr] `username`
```

### DESCRIPTION
The `crontab` utility manages a user’s access with `cron` (see `cron(1M)` by copying, creating, listing, and removing crontab files. If invoked without options, `crontab` copies the specified file, or the standard input if no file is specified, into a directory that holds all users’ crontabs.

If `crontab` is invoked with `filename`, this will overwrite an existing crontab entry for the user that invokes it.

**crontab Access Control**

Users: Access to `crontab` is allowed:
- if the user’s name appears in `/etc/cron.d/cron.allow`.
- if `/etc/cron.d/cron.allow` does not exist and the user’s name is not in `/etc/cron.d/cron.deny`.

Users: Access to `crontab` is denied:
- if `/etc/cron.d/cron.allow` exists and the user’s name is not in it.
- if `/etc/cron.d/cron.allow` does not exist and user’s name is in `/etc/cron.d/cron.deny`.
- if neither file exists, only a user with the `solaris.jobs.user` authorization is allowed to submit a job.

Notice that the rules for `allow` and `deny` apply to `root` only if the `allow/deny` files exist.

The `allow/deny` files consist of one user name per line.

### crontab Entry Format
A crontab file consists of lines of six fields each. The fields are separated by spaces or tabs. The first five are integer patterns that specify the following:

- minute (0−59),
- hour (0−23),
- day of the month (1−31),
- month of the year (1−12),
- day of the week (0−6 with 0=Sunday).

Each of these patterns may be either an asterisk (meaning all legal values) or a list of elements separated by commas. An element is either a number or two numbers separated by a minus sign (meaning an inclusive range). Note that the specification of days may be made by two fields (day of the month and day of the week). Both are adhered to if specified as a list of elements. See EXAMPLES.

The sixth field of a line in a crontab file is a string that is executed by the shell at the specified times. A percent character in this field (unless escaped by `\`) is translated to a `NEWLINE` character.
Only the first line (up to a `#` or end of line) of the command field is executed by the shell. Other lines are made available to the command as standard input. Any line beginning with a `#` is a comment and will be ignored. The file should not contain blank lines.

The shell is invoked from your $HOME directory with an arg0 of sh. Users who desire to have their .profile executed must explicitly do so in the crontab file. cron supplies a default environment for every shell, defining HOME, LOGNAME, SHELL(=/bin/sh), TZ, and PATH. The default PATH for user cron jobs is /usr/bin; while root cron jobs default to /usr/sbin:/usr/bin. The default PATH can be set in /etc/default/cron; see cron(1M).

If you do not redirect the standard output and standard error of your commands, any generated output or errors will be mailed to you.

OPTIONS
The following options are supported:

- `e` Edits a copy of the current user’s crontab file, or creates an empty file to edit if crontab does not exist. When editing is complete, the file is installed as the user’s crontab file. If a username is given, the specified user’s crontab file is edited, rather than the current user’s crontab file; this may only be done by a user with the solaris.jobs.admin authorization. The environment variable EDITOR determines which editor is invoked with the -e option. The default editor is ed(1). Notice that all crontab jobs should be submitted using crontab; you should not add jobs by just editing the crontab file because cron will not be aware of changes made this way.

If all lines in the crontab file are deleted, the old crontab file will be restored. The correct way to delete all lines is to remove the crontab file via the -r option.

- `l` Lists the crontab file for the invoking user. Only a user with the solaris.jobs.admin authorization can specify a username following the -r or -l options to remove or list the crontab file of the specified user.

- `r` Removes a user’s crontab from the crontab directory.

EXAMPLES

**EXAMPLE 1** Cleaning up core files

This example cleans up core files every weekday morning at 3:15 am:

```
15 3 * * 1-5 find $HOME -name core 2>/dev/null | xargs rm -f
```

**EXAMPLE 2** Mailing a birthday greeting

```
0 12 14 2 * mailx john%Happy Birthday!%Time for lunch.
```

**EXAMPLE 3** Specifying days of the month and week

This example
EXAMPLE 3 Specifying days of the month and week (Continued)

```
 0 0 1,15 * 1
```
would run a command on the first and fifteenth of each month, as well as on every Monday.

To specify days by only one field, the other field should be set to *. For example:
```
 0 0 * * 1
```
would run a command only on Mondays.

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of crontab: LC_TYPE, LC_MESSAGES, and NLSPATH.

EDITOR

Determine the editor to be invoked when the -e option is specified. The default editor is ed(1). If both the EDITOR and VISUAL environment variables are set, the value of the VISUAL variable is selected as the editor.

EXIT STATUS

The following exit values are returned:

0  Successful completion.

>0  An error occurred.

FILES

/etc/cron.d  main cron directory
/etc/cron.d/cron.allow  list of allowed users
/etc/default/cron  contains cron default settings
/etc/cron.d/cron.deny  list of denied users
/var/cron/log  accounting information
/var/spool/cron/crontabs  spool area for crontab

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO

atq(1), atrm(1), auths(1), ed(1), sh(1), cron(1M), su(1M), auth_attr(4), attributes(5), environ(5)

NOTES

If you inadvertently enter the crontab command with no argument(s), do not attempt to get out with Control-d. This removes all entries in your crontab file. Instead, exit with Control-c.
If an authorized user modifies another user’s crontab file, resulting behavior may be unpredictable. Instead, the super-user should first use su(1M) to become super-user to the other user’s login before making any changes to the crontab file.

When updating cron, check first for existing crontab entries that may be scheduled close to the time of the update. Such entries may be lost if the update process completes after the scheduled event. This can happen because, when cron is notified by crontab to update the internal view of a user’s crontab file, it first removes the user’s existing internal crontab and any internal scheduled events. Then it reads the new crontab file and rebuilds the internal crontab and events. This last step takes time, especially with a large crontab file, and may complete after an existing crontab entry is scheduled to run if it is scheduled too close to the update. To be safe, start a new job at least 60 seconds after the current date and time.
crypt(1)

NAME  crypt – encode or decode a file
SYNOPSIS  crypt  [password]

DESCRIPTION  The crypt utility encrypts and decrypts the contents of a file. crypt reads from the standard input and writes on the standard output. The password is a key that selects a particular transformation. If no password is given, crypt demands a key from the terminal and turns off printing while the key is being typed in. crypt encrypts and decrypts with the same key:

```bash
example$ crypt key<clear.file> encrypted.file
example$ crypt key<encrypted.file | pr
```

will print the contents of clear.file.

Files encrypted by crypt are compatible with those treated by the editors ed(1), ex(1), and vi(1) in encryption mode.

The security of encrypted files depends on three factors: the fundamental method must be hard to solve; direct search of the key space must be infeasible; “sneak paths” by which keys or cleartext can become visible must be minimized.

crypt implements a one-rotor machine designed along the lines of the German Enigma, but with a 256-element rotor. Methods of attack on such machines are widely known, thus crypt provides minimal security.

The transformation of a key into the internal settings of the machine is deliberately designed to be expensive, that is, to take a substantial fraction of a second to compute. However, if keys are restricted to (say) three lower-case letters, then encrypted files can be read by expending only a substantial fraction of five minutes of machine time.

Since the key is an argument to the crypt command, it is potentially visible to users executing ps(1) or a derivative command. To minimize this possibility, crypt takes care to destroy any record of the key immediately upon entry. No doubt the choice of keys and key security are the most vulnerable aspect of crypt.

FILES  /dev/tty  for typed key

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO  des(1), ed(1), ex(1), makekey(1), ps(1), vi(1), attributes(5)
csh(1)

<table>
<thead>
<tr>
<th>NAME</th>
<th>csh – shell command interpreter with a C-like syntax</th>
</tr>
</thead>
</table>
| SYNOPSIS      | **csh** [-bcefinstvVxx] [argument...]

**DESCRIPTION**

csh, the C shell, is a command interpreter with a syntax reminiscent of the C language. It provides a number of convenient features for interactive use that are not available with the Bourne shell, including filename completion, command aliasing, history substitution, job control, and a number of built-in commands. As with the Bourne shell, the C shell provides variable, command and filename substitution.

**Initialization and Termination**

When first started, the C shell normally performs commands from the `.cshrc` file in your home directory, provided that it is readable and you either own it or your real group ID matches its group ID. If the shell is invoked with a name that starts with ‘-’, as when started by `login(1)`, the shell runs as a login shell.

If the shell is a login shell, this is the sequence of invocations: First, commands in `/etc/.login` are executed. Next, commands from the `.cshrc` file in your home directory are executed. Then the shell executes commands from the `.login` file in your home directory; the same permission checks as those for `.cshrc` are applied to this file. Typically, the `.login` file contains commands to specify the terminal type and environment. (For an explanation of file interpreters, see below 'Command Execution' and `exec(2)`.)

As a login shell terminates, it performs commands from the `.logout` file in your home directory; the same permission checks as those for `.cshrc` are applied to this file.

**Interactive Operation**

After startup processing is complete, an interactive C shell begins reading commands from the terminal, prompting with `hostname%` (or `hostname#` for the privileged user). The shell then repeatedly performs the following actions: a line of command input is read and broken into words. This sequence of words is placed on the history list and then parsed, as described under USAGE, below. Finally, the shell executes each command in the current line.

**Noninteractive Operation**

When running noninteractively, the shell does not prompt for input from the terminal. A noninteractive C shell can execute a command supplied as an `argument` on its command line, or interpret commands from a file, also known as a script.

**OPTIONS**

The following options are supported:

- `-b` Force a “break” from option processing. Subsequent command line arguments are not interpreted as C shell options. This allows the passing of options to a script without confusion. The shell does not run set-user-ID or set-group-ID scripts unless this option is present.

- `-c` Execute the first `argument` (which must be present). Remaining arguments are placed in `argv`, the argument-list variable, and passed directly to `csh`.

- `-e` Exit if a command terminates abnormally or yields a nonzero exit status.
-f  Fast start. Read neither the .cshrc file, nor the .login file (if a login shell) upon startup.

-i  Forced interactive. Prompt for command line input, even if the standard input does not appear to be a terminal (character-special device).

-n  Parse (interpret), but do not execute commands. This option can be used to check C shell scripts for syntax errors.

-s  Take commands from the standard input.

-t  Read and execute a single command line. A '\' (backslash) can be used to escape each newline for continuation of the command line onto subsequent input lines.

-v  Verbose. Set the verbose predefined variable; command input is echoed after history substitution (but before other substitutions) and before execution.

-V  Set verbose before reading .cshrc.

-x  Echo. Set the echo variable; echo commands after all substitutions and just before execution.

-X  Set echo before reading .cshrc.

Except with the options -c, -i, -s, or -t, the first nonoption argument is taken to be the name of a command or script. It is passed as argument zero, and subsequent arguments are added to the argument list for that command or script.

When enabled by setting the variable filec, an interactive C shell can complete a partially typed filename or user name. When an unambiguous partial filename is followed by an ESC character on the terminal input line, the shell fills in the remaining characters of a matching filename from the working directory.

If a partial filename is followed by the EOF character (usually typed as CTRL-d), the shell lists all filenames that match. It then prompts once again, supplying the incomplete command line typed in so far.

When the last (partial) word begins with a tilde (~), the shell attempts completion with a user name, rather than a file in the working directory.

The terminal bell signals errors or multiple matches; this can be inhibited by setting the variable nobeep. You can exclude files with certain suffixes by listing those suffixes in the variable fignore. If, however, the only possible completion includes a suffix in the list, it is not ignored. fignore does not affect the listing of filenames by the EOF character.
Lexical Structure

The shell splits input lines into words at space and tab characters, except as noted below. The characters & | ; < > ( and ) form separate words; if paired, the pairs form single words. These shell metacharacters can be made part of other words, and their special meaning can be suppressed by preceding them with a ‘\’ (backslash). A newline preceded by a \ is equivalent to a space character.

In addition, a string enclosed in matched pairs of single-quotes ( ’ ), double-quotes ( " ), or backquotes ( ‘ ) forms a partial word; metacharacters in such a string, including any space or tab characters, do not form separate words. Within pairs of backquote (’ ') or double-quote (" " ) characters, a newline preceded by a ‘\’ (backslash) gives a true newline character. Additional functions of each type of quote are described, below, under Variable Substitution, Command Substitution, and Filename Substitution.

When the shell’s input is not a terminal, the character # introduces a comment that continues to the end of the input line. Its special meaning is suppressed when preceded by a \ or enclosed in matching quotes.

Command Line Parsing

A simple command is composed of a sequence of words. The first word (that is not part of an I/O redirection) specifies the command to be executed. A simple command, or a set of simple commands separated by | or & characters, forms a pipeline. With |, the standard output of the preceding command is redirected to the standard input of the command that follows. With | & , both the standard error and the standard output are redirected through the pipeline.

Pipelines can be separated by semicolons (; ), in which case they are executed sequentially. Pipelines that are separated by && or | | form conditional sequences in which the execution of pipelines on the right depends upon the success or failure, respectively, of the pipeline on the left.

A pipeline or sequence can be enclosed within parentheses (’’) to form a simple command that can be a component in a pipeline or sequence.

A sequence of pipelines can be executed asynchronously or “in the background” by appending an ‘&’; rather than waiting for the sequence to finish before issuing a prompt, the shell displays the job number (see Job Control, below) and associated process IDs and prompts immediately.

History Substitution

History substitution allows you to use words from previous command lines in the command line you are typing. This simplifies spelling corrections and the repetition of complicated commands or arguments. Command lines are saved in the history list, the size of which is controlled by the history variable. The most recent command is retained in any case. A history substitution begins with a ! (although you can change this with the histchars variable) and may occur anywhere on the command line; history substitutions do not nest. The ! can be escaped with \ to suppress its special meaning.
Input lines containing history substitutions are echoed on the terminal after being expanded, but before any other substitutions take place or the command gets executed.

### Event Designators

An event designator is a reference to a command line entry in the history list.

<table>
<thead>
<tr>
<th>Designator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>!</code></td>
<td>Start a history substitution, except when followed by a space character, tab, newline, = or <code>(</code>.</td>
</tr>
<tr>
<td><code>!!</code></td>
<td>Refer to the previous command. By itself, this substitution repeats the previous command.</td>
</tr>
<tr>
<td><code>!n</code></td>
<td>Refer to command line <code>n</code>.</td>
</tr>
<tr>
<td><code>!-n</code></td>
<td>Refer to the current command line minus <code>n</code>.</td>
</tr>
<tr>
<td><code>!str</code></td>
<td>Refer to the most recent command starting with <code>str</code>.</td>
</tr>
<tr>
<td><code>!?str?</code></td>
<td>Refer to the most recent command containing <code>str</code>.</td>
</tr>
<tr>
<td><code>!?str? additional</code></td>
<td>Refer to the most recent command containing <code>str</code> and append <code>additional</code> to that referenced command.</td>
</tr>
<tr>
<td><code>!{command} additional</code></td>
<td>Refer to the most recent command beginning with <code>command</code> and append <code>additional</code> to that referenced command.</td>
</tr>
<tr>
<td><code>^previous_word^replacement^</code></td>
<td>Repeat the previous command line replacing the string <code>previous_word</code> with the string <code>replacement</code>. This is equivalent to the history substitution: <code>!:u/previous_word/replacement/</code>.</td>
</tr>
</tbody>
</table>

To re-execute a specific previous command AND make such a substitution, say, re-executing command #6, `!:6s/previous_word/replacement/`.

### Word Designators

A `:` (colon) separates the event specification from the word designator. It can be omitted if the word designator begins with a `^`, `$`, `*`, `−`, or `%`. If the word is to be selected from the previous command, the second `!` character can be omitted from the event specification. For instance, `!!!2` and `!2` both refer to the first word of the previous command, while `!!$` and `$` both refer to the last word in the previous command. Word designators include:
# The entire command line typed so far.
0 The first input word (command).
\(n\) The \(n\)'th argument.
^ The first argument, that is, 1.
$ The last argument.
% The word matched by (the most recent) ?s search.
\(x-y\) A range of words; \(-y\) abbreviates \(0-y\).
* All the arguments, or a null value if there is just one word in the event.
\(x^*\) Abbreviates \(x-\$\).
\(x^-\) Like \(x^*\) but omitting word $.

### Modifiers
After the optional word designator, you can add one of the following modifiers, preceded by a :.

\(h\) Remove a trailing pathname component, leaving the head.
\(r\) Remove a trailing suffix of the form '.xxx', leaving the basename.
\(e\) Remove all but the suffix, leaving the Extension.
\(s/l/r/\) Substitute \(r\) for \(l\).
\(t\) Remove all leading pathname components, leaving the tail.
\(\&\) Repeat the previous substitution.
\(g\) Apply the change to the first occurrence of a match in each word, by prefixing the above (for example, \(g\&\)).
\(p\) Print the new command but do not execute it.
\(q\) Quote the substituted words, escaping further substitutions.
\(x\) Like \(q\), but break into words at each space character, tab or newline.

Unless preceded by a \(g\), the modification is applied only to the first string that matches \(l\); an error results if no string matches.

The left-hand side of substitutions are not regular expressions, but character strings. Any character can be used as the delimiter in place of / . A backslash quotes the delimiter character. The character \&\(s\), in the right hand side, is replaced by the text from the left-hand-side. The \& can be quoted with a backslash. A null l uses the previous string either from a \(l\) or from a contextual scan string \(s\) from !?s. You can omit the rightmost delimiter if a newline immediately follows \(r\); the rightmost ? in a context scan can similarly be omitted.

Without an event specification, a history reference refers either to the previous command, or to a previous history reference on the command line (if any).
Quick Substitution

This is equivalent to the history substitution: \!s/l/r/.

Aliases

The C shell maintains a list of aliases that you can create, display, and modify using the alias and unalias commands. The shell checks the first word in each command to see if it matches the name of an existing alias. If it does, the command is reprocessed with the alias definition replacing its name; the history substitution mechanism is made available as though that command were the previous input line. This allows history substitutions, escaped with a backslash in the definition, to be replaced with actual command line arguments when the alias is used. If no history substitution is called for, the arguments remain unchanged.

Aliases can be nested. That is, an alias definition can contain the name of another alias. Nested aliases are expanded before any history substitutions is applied. This is useful in pipelines such as

```
alias lm 'ls -l \!* | more'
```

which when called, pipes the output of `ls(1)` through `more(1)`.

Except for the first word, the name of the alias may not appear in its definition, nor in any alias referred to by its definition. Such loops are detected, and cause an error message.

I/O Redirection

The following metacharacters indicate that the subsequent word is the name of a file to which the command’s standard input, standard output, or standard error is redirected; this word is variable, command, and filename expanded separately from the rest of the command.

```
< Redirect the standard input.
< < word

Read the standard input, up to a line that is identical with `word`, and place the resulting lines in a temporary file. Unless `word` is escaped or quoted, variable and command substitutions are performed on these lines. Then, the pipeline is invoked with the temporary file as its standard input. `word` is not subjected to variable, filename, or command substitution, and each line is compared to it before any substitutions are performed by the shell.

> >! >& >&!

Redirect the standard output to a file. If the file does not exist, it is created. If it does exist, it is overwritten; its previous contents are lost.

When set, the variable noclobber prevents destruction of existing files. It also prevents redirection to terminals and `/dev/null`, unless one of the \! forms is used. The & forms redirect both standard output and the standard error (diagnostic output) to the file.

```
> > > >& >! > >&!

Append the standard output. Like `>`, but places output at the end of the file rather than overwriting it. If noclobber is set, it is an error for the file not to exist, unless
one of the ! forms is used. The & forms append both the standard error and standard output to the file.

The C shell maintains a set of variables, each of which is composed of a name and a value. A variable name consists of up to 20 letters and digits, and starts with a letter (the underscore is considered a letter). A variable’s value is a space-separated list of zero or more words.

To refer to a variable’s value, precede its name with a ‘$’. Certain references (described below) can be used to select specific words from the value, or to display other information about the variable. Braces can be used to insulate the reference from other characters in an input-line word.

Variable substitution takes place after the input line is analyzed, aliases are resolved, and I/O redirections are applied. Exceptions to this are variable references in I/O redirections (substituted at the time the redirection is made), and backquoted strings (see Command Substitution).

Variable substitution can be suppressed by preceding the $ with a \, except within double-quotes where it always occurs. Variable substitution is suppressed inside of single-quotes. A $ is escaped if followed by a space character, tab or newline.

Variables can be created, displayed, or destroyed using the set and unset commands. Some variables are maintained or used by the shell. For instance, the argv variable contains an image of the shell’s argument list. Of the variables used by the shell, a number are toggles; the shell does not care what their value is, only whether they are set or not.

Numerical values can be operated on as numbers (as with the @ built-in command). With numeric operations, an empty value is considered to be zero; the second and subsequent words of multiword values are ignored. For instance, when the verbose variable is set to any value (including an empty value), command input is echoed on the terminal.

Command and filename substitution is subsequently applied to the words that result from the variable substitution, except when suppressed by double-quotes, when noglob is set (suppressing filename substitution), or when the reference is quoted with the :q modifier. Within double-quotes, a reference is expanded to form (a portion of) a quoted string; multiword values are expanded to a string with embedded space characters. When the :q modifier is applied to the reference, it is expanded to a list of space-separated words, each of which is quoted to prevent subsequent command or filename substitutions.

Except as noted below, it is an error to refer to a variable that is not set.

$var
$\{var\}

These are replaced by words from the value of var, each separated by a space character. If var is an environment variable, its value is returned (but ‘:’ modifiers and the other forms given below are not available).
These select only the indicated words from the value of `var`. Variable substitution is applied to `index`, which may consist of (or result in) a either single number, two numbers separated by a ‘−’, or an asterisk. Words are indexed starting from 1; a ‘*’ selects all words. If the first number of a range is omitted (as with `$argv[-2]`), it defaults to 1. If the last number of a range is omitted (as with `$argv[1-]`), it defaults to $#var (the word count). It is not an error for a range to be empty if the second argument is omitted (or within range).

These give the number of words in the variable.

This substitutes the name of the file from which command input is being read except for setuid shell scripts. An error occurs if the name is not known.

Equivalent to `$argv[n]`.

Equivalent to `$argv[*]`.

The modifiers :e, :h, :q, :r, :t, and :x can be applied (see History Substitution), as can :gh, :gt, and :gr. If {} (braces) are used, then the modifiers must appear within the braces. The current implementation allows only one such modifier per expansion.

The following references may not be modified with : modifiers.

Substitutes the string 1 if `var` is set or 0 if it is not set.

Substitutes 1 if the current input filename is known or 0 if it is not.

Substitutes the process number of the (parent) shell.

Substitutes a line from the standard input, with no further interpretation thereafter. It can be used to read from the keyboard in a C shell script.

Command and filename substitutions are applied selectively to the arguments of built-in commands. Portions of expressions that are not evaluated are not expanded. For non-built-in commands, filename expansion of the command name is done separately from that of the argument list; expansion occurs in a subshell, after I/O redirection is performed.

A command enclosed by backquotes (' ... ') is performed by a subshell. Its standard output is broken into separate words at each space character, tab and newline; null words are discarded. This text replaces the backquoted string on the current command.
Filename Substitution

Unquoted words containing any of the characters *, ?, [ or {, or that begin with ~, are expanded (also known as globbing) to an alphabetically sorted list of filenames, as follows:

*          Match any (zero or more) characters.
?          Match any single character.
[..]       Match any single character in the enclosed list(s) or range(s). A list is a string of characters. A range is two characters separated by a dash (–), and includes all the characters in between in the ASCII collating sequence (see ascii(5)).
{ str, str, ... } Expand to each string (or filename-matching pattern) in the comma-separated list. Unlike the pattern-matching expressions above, the expansion of this construct is not sorted. For instance, {b,a} expands to 'b' 'a', (not 'a' 'b'). As special cases, the characters { and }, along with the string { }, are passed undisturbed.
~[user]    Your home directory, as indicated by the value of the variable home, or that of user, as indicated by the password entry for user.

Expressions and Operators

Only the patterns *, ?, and [...] imply pattern matching; an error results if no filename matches a pattern that contains them. The ‘.’ (dot character), when it is the first character in a filename or pathname component, must be matched explicitly. The / (slash) must also be matched explicitly.

A number of C shell built-in commands accept expressions, in which the operators are similar to those of C and have the same precedence. These expressions typically appear in the @, exit, if, set and while commands, and are often used to regulate the flow of control for executing commands. Components of an expression are separated by white space.

Null or missing values are considered 0. The result of all expressions is a string, which may represent decimal numbers.

The following C shell operators are grouped in order of precedence:

( ... )      grouping
>~            one’s complement
!             logical negation
* / %         multiplication, division, remainder (these are right associative, which can lead to unexpected results; group combinations explicitly with parentheses.)
The operators: ==, !=, =~, and !~ compare their arguments as strings; other operators use numbers. The operators == and != each check whether or not a string to the left matches a filename substitution pattern on the right. This reduces the need for switch statements when pattern-matching between strings is all that is required.

Also available are file inquiries:

- **-r filename** 
  Return true, or 1 if the user has read access. Otherwise it returns false, or 0.

- **-w filename** 
  True if the user has write access.

- **-x filename** 
  True if the user has execute permission (or search permission on a directory).

- **-e filename** 
  True if filename exists.

- **-o filename** 
  True if the user owns filename.

- **-z filename** 
  True if filename is of zero length (empty).

- **-f filename** 
  True if filename is a plain file.

- **-d filename** 
  True if filename is a directory.

If filename does not exist or is inaccessible, then all inquiries return false.

An inquiry as to the success of a command is also available:

{ command } 
If command runs successfully, the expression evaluates to true, 1. Otherwise, it evaluates to false, 0. (Note: Conversely, command itself typically returns 0 when it runs successfully, or some other value if it encounters a problem. If you want to get at the status directly, use the value of the status variable rather than this expression).
Control Flow

The shell contains a number of commands to regulate the flow of control in scripts and within limits, from the terminal. These commands operate by forcing the shell either to reread input (to loop), or to skip input under certain conditions (to branch).

Each occurrence of a foreach, switch, while, if...then and else built-in command must appear as the first word on its own input line.

If the shell’s input is not seekable and a loop is being read, that input is buffered. The shell performs seeks within the internal buffer to accomplish the rereading implied by the loop. (To the extent that this allows, backward goto commands will succeed on nonseekable inputs.)

Command Execution

If the command is a C shell built-in command, the shell executes it directly. Otherwise, the shell searches for a file by that name with execute access. If the command name contains a /, the shell takes it as a pathname, and searches for it. If the command name does not contain a /, the shell attempts to resolve it to a pathname, searching each directory in the path variable for the command. To speed the search, the shell uses its hash table (see the rehash built-in command) to eliminate directories that have no applicable files. This hashing can be disabled with the -c or -t, options, or the unhash built-in command.

As a special case, if there is no / in the name of the script and there is an alias for the word shell, the expansion of the shell alias is prepended (without modification) to the command line. The system attempts to execute the first word of this special (late-occurring) alias, which should be a full pathname. Remaining words of the alias’s definition, along with the text of the input line, are treated as arguments.

When a pathname is found that has proper execute permissions, the shell forks a new process and passes it along with its arguments, to the kernel using the execve() system call (see exec(2)). The kernel then attempts to overlay the new process with the desired program. If the file is an executable binary (in a.out(4) format) the kernel succeeds and begins executing the new process. If the file is a text file and the first line begins with #!, the next word is taken to be the pathname of a shell (or command) to interpret that script. Subsequent words on the first line are taken as options for that shell. The kernel invokes (overlays) the indicated shell, using the name of the script as an argument.

If neither of the above conditions holds, the kernel cannot overlay the file and the execve( ) call fails (see exec(2)); the C shell then attempts to execute the file by spawning a new shell, as follows:

- If the first character of the file is a #, a C shell is invoked.
- Otherwise, a Bourne shell is invoked.

Signal Handling

The shell normally ignores QUIT signals. Background jobs are immune to signals generated from the keyboard, including hangups (HUP). Other signals have the values that the C shell inherited from its environment. The shell’s handling of interrupt and

User Commands 231
terminate signals within scripts can be controlled by the `onintr` built-in command. Login shells catch the TERM signal; otherwise, this signal is passed on to child processes. In no case are interrupts allowed when a login shell is reading the `.logout` file.

### Job Control

The shell associates a numbered `job` with each command sequence to keep track of those commands that are running in the background or have been stopped with TSTP signals (typically CTRL-z). When a command or command sequence (semicolon separated list) is started in the background using the `&` metacharacter, the shell displays a line with the job number in brackets and a list of associated process numbers:

```
[1] 1234
```

To see the current list of jobs, use the `jobs` built-in command. The job most recently stopped (or put into the background if none are stopped) is referred to as the `current` job and is indicated with a `+`. The previous job is indicated with a `−`; when the current job is terminated or moved to the foreground, this job takes its place (becomes the new current job).

To manipulate jobs, refer to the `bg`, `fg`, `kill`, `stop`, and `%` built-in commands.

A reference to a job begins with a `%`. By itself, the percent-sign refers to the current job.

- `% %+ %%` The current job.
- `%−` The previous job.
- `%j` Refer to job `j` as in: `kill -9 %j`. `j` can be a job number, or a string that uniquely specifies the command line by which it was started; `fg %vi` might bring a stopped `vi` job to the foreground, for instance.
- `%?string` Specify the job for which the command line uniquely contains `string`.

A job running in the background stops when it attempts to read from the terminal. Background jobs can normally produce output, but this can be suppressed using the `stty tostop` command.

### Status Reporting

While running interactively, the shell tracks the status of each job and reports whenever the job finishes or becomes blocked. It normally displays a message to this effect as it issues a prompt, in order to avoid disturbing the appearance of your input. When set, the `notify` variable indicates that the shell is to report status changes immediately. By default, the `notify` command marks the current process; after starting a background job, type `notify` to mark it.

### Commands

Built-in commands are executed within the C shell. If a built-in command occurs as any component of a pipeline except the last, it is executed in a subshell.
Null command. This command is interpreted, but performs no action.

**alias [ name [ def ]]**

Assign *def* to the alias *name*. *def* is a list of words that may contain escaped history-substitution metasyntax. *name* is not allowed to be *alias* or *unalias*. If *def* is omitted, the current definition for the alias *name* is displayed. If both *name* and *def* are omitted, all aliases are displayed with their definitions.

**bg [ $job ...]**

Run the current or specified jobs in the background.

**break**

Resume execution after the end of the nearest enclosing *foreach* or *while* loop. The remaining commands on the current line are executed. This allows multilevel breaks to be written as a list of *break* commands, all on one line.

**breaksw**

Break from a *switch*, resuming after the *endsw*.

**case label :**

A label in a *switch* statement.

**cd [ dir ]**

**chdir [ dir ]**

Change the shell’s working directory to directory *dir*. If no argument is given, change to the home directory of the user. If *dir* is a relative pathname not found in the current directory, check for it in those directories listed in the *cdpath* variable. If *dir* is the name of a shell variable whose value starts with a `/`, change to the directory named by that value.

**continue**

Continue execution of the next iteration of the nearest enclosing *while* or *foreach* loop.

**default :**

Labels the default case in a *switch* statement. The default should come after all *case* labels. Any remaining commands on the command line are first executed.

**dirs [-l]**

Print the directory stack, most recent to the left; the first directory shown is the current directory. With the `-l` argument, produce an unabbreviated printout; use of the `~` notation is suppressed.

**echo [-n] list**

The words in *list* are written to the shell’s standard output, separated by space characters. The output is terminated with a newline unless the `-n` option is used. *csh* will, by default, invoke its built-in *echo*, if *echo* is called without the full pathname of a Unix command, regardless of the configuration of your *PATH* (see *echo*(1)).
**eval** `argument` ...

Reads the arguments as input to the shell and executes the resulting command(s). This is usually used to execute commands generated as the result of command or variable substitution. See `tset(1B)` for an example of how to use `eval`.

**exec** `command`

Execute `command` in place of the current shell, which terminates.

**exit** `[ (expr) ]`

The calling shell or shell script exits, either with the value of the status variable or with the value specified by the expression `expr`.

**fg** `%job`

Bring the current or specified `job` into the foreground.

**foreach** `var` `(wordlist)`

... 

end

The variable `var` is successively set to each member of `wordlist`. The sequence of commands between this command and the matching `end` is executed for each new value of `var`. Both `foreach` and `end` must appear alone on separate lines.

The built-in command `continue` may be used to terminate the execution of the current iteration of the loop and the built-in command `break` may be used to terminate execution of the `foreach` command. When this command is read from the terminal, the loop is read once prompting with `?` before any statements in the loop are executed.

**glob** `wordlist`

Perform filename expansion on `wordlist`. Like `echo`, but no `\` escapes are recognized. Words are delimited by `NULL` characters in the output.

**goto** `label`

The specified `label` is a filename and a command expanded to yield a label. The shell rewinds its input as much as possible and searches for a line of the form `label:` possibly preceded by space or tab characters. Execution continues after the indicated line. It is an error to jump to a label that occurs between a `while` or `for` built-in command and its corresponding `end`.

**hashstat**

Print a statistics line indicating how effective the internal hash table for the `path` variable has been at locating commands (and avoiding `execs`). An `exec` is attempted for each component of the `path` where the hash function indicates a possible hit and in each component that does not begin with a `'/'.` These statistics only reflect the effectiveness of the `path` variable, not the `cdpath` variable.

**history** `[-hr] [ n ]`

Display the history list; if `n` is given, display only the `n` most recent events.

`-r` Reverse the order of printout to be most recent first rather than oldest first.
-h Display the history list without leading numbers. This is used to produce files suitable for sourcing using the -h option to source.

if (expr) command
If the specified expression evaluates to true, the single command with arguments is executed. Variable substitution on command happens early, at the same time it does for the rest of the if command. command must be a simple command, not a pipeline, a command list, or a parenthesized command list. Note: I/O redirection occurs even if expr is false, when command is not executed (this is a bug).

if (expr) then
... else if (expr2) then
... else
...
endif
If expr is true, commands up to the first else are executed. Otherwise, if expr2 is true, the commands between the else if and the second else are executed. Otherwise, commands between the else and the endif are executed. Any number of else if pairs are allowed, but only one else. Only one endif is needed, but it is required. The words else and endif must be the first nonwhite characters on a line. The if must appear alone on its input line or after an else.

jobs [-l]
List the active jobs under job control.

-1 List process IDs, in addition to the normal information.

kill [ -sig ] [ pid ] [ %job ] ...
kill -l
Send the TERM (terminate) signal, by default, or the signal specified, to the specified process ID, the job indicated, or the current job. Signals are either given by number or by name. There is no default. Typing kill does not send a signal to the current job. If the signal being sent is TERM (terminate) or HUP (hangup), then the job or process is sent a CONT (continue) signal as well.

-1 List the signal names that can be sent.

limit [-h] [resource [max-use ]]
Limit the consumption by the current process or any process it spawns, each not to exceed max-use on the specified resource. If max-use is omitted, print the current limit; if resource is omitted, display all limits. (Run the sysdef(1M) command to obtain the maximum possible limits for your system. The values reported are in hexadecimal, but can be translated into decimal numbers using the bc(1) command).

-h Use hard limits instead of the current limits. Hard limits impose a ceiling on the values of the current limits. Only the privileged user may raise the hard limits.

resource is one of:
cputime Maximum CPU seconds per process.
filesize Largest single file allowed; limited to the size of the filesystem. (see df(1M)).
datasize (heapsize) Maximum data size (including stack) for the process. This is the size of your virtual memory. See swap(1M).
stacksize Maximum stack size for the process. See swap(1M).
coredumpsize Maximum size of a core dump (file). This limited to the size of the filesystem.
descriptors Maximum number of file descriptors. Run sysdef().
memorysize Maximum size of virtual memory.

max-use is a number, with an optional scaling factor, as follows:

\textit{nh} \quad \text{Hours (for cputime).}
\textit{nk} \quad n \text{ kilobytes. This is the default for all but cputime.}
\textit{nm} \quad n \text{ megabytes or minutes (for cputime).}
\textit{mm:ss} \quad \text{Minutes and seconds (for cputime).}

Example of limit: to limit the size of a core file dump to 0 Megabytes, type the following:

\texttt{limit coredumpsize 0M}

login [\textit{username} | -p ]
   \text{Terminate a login shell and invoke login(1). The .logout file is not processed. If username is omitted, login prompts for the name of a user.}
\texttt{-p} \quad \text{Preserve the current environment (variables).}

logout
   \text{Terminate a login shell.}

nice [+n | -n ] [\textit{command} ]
   \text{Increment the process priority value for the shell or for \textit{command} by \textit{n}. The higher the priority value, the lower the priority of a process, and the slower it runs. When given, \textit{command} is always run in a subshell, and the restrictions placed on commands in simple if commands apply. If \textit{command} is omitted, nice increments the value for the current shell. If no increment is specified, nice sets the process priority value to 4. The range of process priority values is from \textdaggerhyphen{20} to 20. Values of \textit{n} outside this range set the value to the lower, or to the higher boundary, respectively.}
\texttt{+n} \quad \text{Increment the process priority value by \textit{n}.}
\texttt{-n} \quad \text{Decrement by \textit{n}. This argument can be used only by the privileged user.}
nohup [command ]
Run command with HUPs ignored. With no arguments, ignore HUPs throughout the remainder of a script. When given, command is always run in a subshell, and the restrictions placed on commands in simple if statements apply. All processes detached with & are effectively nohup’d.

notify [%job]...
Notify the user asynchronously when the status of the current job or specified jobs changes.

onintr [− | label]
Control the action of the shell on interrupts. With no arguments, onintr restores the default action of the shell on interrupts. (The shell terminates shell scripts and returns to the terminal command input level). With the − argument, the shell ignores all interrupts. With a label argument, the shell executes a goto label when an interrupt is received or a child process terminates because it was interrupted.

popd [ +n ]
Pop the directory stack and cd to the new top directory. The elements of the directory stack are numbered from 0 starting at the top.

+ n Discard the n’th entry in the stack.

pushd [ +n | dir]
Push a directory onto the directory stack. With no arguments, exchange the top two elements.

+ n Rotate the n’th entry to the top of the stack and cd to it.

dir Push the current working directory onto the stack and change to dir.

rehash
Recompute the internal hash table of the contents of directories listed in the path variable to account for new commands added. Recompute the internal hash table of the contents of directories listed in the cdpath variable to account for new directories added.

repeat count command
Repeat command count times. command is subject to the same restrictions as with the one-line if statement.

set [ var [= value ] ]
set var [n] = word
With no arguments, set displays the values of all shell variables. Multiword values are displayed as a parenthesized list. With the var argument alone, set assigns an empty (null) value to the variable var. With arguments of the form var = value set assigns value to var, where value is one of:

word A single word (or quoted string).
(wordlist) A space-separated list of words enclosed in parentheses.

Values are command and filename expanded before being assigned. The form set var [n] = word replaces the n’th word in a multiword value with word.
setenv [VAR [word ]]

With no arguments, setenv displays all environment variables. With the VAR argument, setenv sets the environment variable VAR to have an empty (null) value. (By convention, environment variables are normally given upper-case names.) With both VAR and word arguments, setenv sets the environment variable NAME to the value word, which must be either a single word or a quoted string. The most commonly used environment variables, USER, TERM, and PATH, are automatically imported to and exported from the csh variables user, term, and path; there is no need to use setenv for these. In addition, the shell sets the PWD environment variable from the csh variable cwd whenever the latter changes.

The environment variables LC_CTYPE, LC_MESSAGES, LC_TIME, LC_COLLATE, LC_NUMERIC, and LC_MONETARY take immediate effect when changed within the C shell.

If any of the LC_* variables (LC_CTYPE, LC_MESSAGES, LC_TIME, LC_COLLATE, LC_NUMERIC, and LC_MONETARY) (see environ(5)) are not set in the environment, the operational behavior of csh for each corresponding locale category is determined by the value of the LANG environment variable. If LC_ALL is set, its contents are used to override both the LANG and the other LC_* variables. If none of the above variables is set in the environment, the "C" (U.S. style) locale determines how csh behaves.

LC_CTYPE Determines how csh handles characters. When LC_CTYPE is set to a valid value, csh can display and handle text and filenames containing valid characters for that locale.

LC_MESSAGES Determines how diagnostic and informative messages are presented. This includes the language and style of the messages and the correct form of affirmative and negative responses. In the "C" locale, the messages are presented in the default form found in the program itself (in most cases, U.S./English).

LC_NUMERIC Determines the value of the radix character (decimal point (".") in the "C" locale) and thousand separator (empty string (""") in the "C" locale).

shift [variable ]

The components of argv, or variable, if supplied, are shifted to the left, discarding the first component. It is an error for the variable not to be set or to have a null value.

source [-h] name

Reads commands from name. source commands may be nested, but if they are nested too deeply the shell may run out of file descriptors. An error in a sourced file at any level terminates all nested source commands.

- h Place commands from the file name on the history list without executing them.
stop %jobid ...  
Stop the current or specified background job.

stop pid ...  
Stop the specified process, pid. (see ps(1)).

suspend  
Stop the shell in its tracks, much as if it had been sent a stop signal with ^Z. This is most often used to stop shells started by su.

switch (string)  
case label:  
...  
breaksw  
...  
default:  
...  
breaksw  
endsw  
Each label is successively matched, against the specified string, which is first command and filename expanded. The file metacharacters *?, and [...] may be used in the case labels, which are variable expanded. If none of the labels match before a "default" label is found, execution begins after the default label. Each case statement and the default statement must appear at the beginning of a line. The command breaksw continues execution after the endsw. Otherwise control falls through subsequent case and default statements as with C. If no label matches and there is no default, execution continues after the endsw.

time [command ]  
With no argument, print a summary of time used by this C shell and its children. With an optional command, execute command and print a summary of the time it uses. As of this writing, the time built-in command does NOT compute the last 6 fields of output, rendering the output to erroneously report the value "0" for these fields.

    example $time ls -R
    9.0u 11.0s 3:32 10% 0+0k 0+0io 0pf+0w

(See below the “Environment Variables and Predefined Shell Variables” sub-section on the time variable.)

umask [value ]  
Display the file creation mask. With value, set the file creation mask. With value given in octal, the user can turn-off any bits, but cannot turn-on bits to allow new permissions. Common values include 077, restricting all permissions from everyone else; 002, giving complete access to the group, and read (and directory search) access to others; or 022, giving read (and directory search) but not write permission to the group and others.

unalias pattern  
Discard aliases that match (filename substitution) pattern. All aliases are removed by ‘unalias *’.
csh(1)

unhash
  Disable the internal hash tables for the path and cdpath variables.

unlimit [-h] [resource]
  Remove a limitation on resource. If no resource is specified, then all resource
  limitations are removed. See the description of the limit command for the list of
  resource names.
  -h       Remove corresponding hard limits. Only the privileged user may do
           this.

unset pattern
  Remove variables whose names match (filename substitution) pattern. All variables
  are removed by ‘unset *’; this has noticeably distasteful side effects.

unsetenv variable
  Remove variable from the environment. As with unset, pattern matching is not
  performed.

wait
  Wait for background jobs to finish (or for an interrupt) before prompting.

while (expr)
...
end
  While expr is true (evaluates to nonzero), repeat commands between the while
  and the matching end statement. break and continue may be used to terminate or
  continue the loop prematurely. The while and end must appear alone on their
  input lines. If the shell’s input is a terminal, it prompts for commands with a
  question-mark until the end command is entered and then performs the commands
  in the loop.

% [job] [%]
  Bring the current or indicated job to the foreground. With the ampersand, continue
  running job in the background.

@ [var = expr]
@ [var[n] = expr]
  With no arguments, display the values for all shell variables. With arguments, set
  the variable var, or the n’th word in the value of var, to the value that expr evaluates
  to. (If [n] is supplied, both var and its n’th component must already exist.)

  If the expression contains the characters >, <, &, or |, then at least this part of expr
  must be placed within parentheses.

  The operators *=, +=, and so forth, are available as in C. The space separating the
  name from the assignment operator is optional. Spaces are, however, mandatory in
  separating components of expr that would otherwise be single words.

  Special postfix operators, + + and − −, increment or decrement name, respectively.
Unlike the Bourne shell, the C shell maintains a distinction between environment variables, which are automatically exported to processes it invokes, and shell variables, which are not. Both types of variables are treated similarly under variable substitution. The shell sets the variables `argv`, `cwd`, `home`, `path`, `prompt`, `shell`, and `status` upon initialization. The shell copies the environment variable `USER` into the shell variable `user`, `TERM` into `term`, and `HOME` into `home`, and copies each back into the respective environment variable whenever the shell variables are reset. `PATH` and `path` are similarly handled. You need only set `path` once in the `.cshrc` or `.login` file. The environment variable `PWD` is set from `cwd` whenever the latter changes. The following shell variables have predefined meanings:

- **argv**: Argument list. Contains the list of command line arguments supplied to the current invocation of the shell. This variable determines the value of the positional parameters `$1`, `$2`, and so on.
- **cdpath**: Contains a list of directories to be searched by the `cd`, `chdir`, and `popd` commands, if the directory argument each accepts is not a subdirectory of the current directory.
- **cwd**: The full pathname of the current directory.
- **echo**: Echo commands (after substitutions) just before execution.
- **fignore**: A list of filename suffixes to ignore when attempting filename completion. Typically the single word `.o`.
- **filec**: Enable filename completion, in which case the CTRL-d character EOT and the ESC character have special significance when typed in at the end of a terminal input line:
  - EOT: Print a list of all filenames that start with the preceding string.
  - ESC: Replace the preceding string with the longest unambiguous extension.
- **hardpaths**: If set, pathnames in the directory stack are resolved to contain no symbolic-link components.
- **histchars**: A two-character string. The first character replaces `!` as the history-substitution character. The second replaces the carat (`^`) for quick substitutions.
- **history**: The number of lines saved in the history list. A very large number may use up all of the C shell's memory. If not set, the C shell saves only the most recent command.
- **home**: The user's home directory. The filename expansion of `~` refers to the value of this variable.
- **ignoreeof**: If set, the shell ignores EOF from terminals. This protects against accidentally killing a C shell by typing a CTRL-d.
mail
A list of files where the C shell checks for mail. If the first word of the value is a number, it specifies a mail checking interval in seconds (default 5 minutes).

nobeep
Suppress the bell during command completion when asking the C shell to extend an ambiguous filename.

noclobber
Restrict output redirection so that existing files are not destroyed by accident. > redirections can only be made to new files. >> redirections can only be made to existing files.

noglob
Inhibit filename substitution. This is most useful in shell scripts once filenames (if any) are obtained and no further expansion is desired.

nonomatch
Returns the filename substitution pattern, rather than an error, if the pattern is not matched. Malformed patterns still result in errors.

notify
If set, the shell notifies you immediately as jobs are completed, rather than waiting until just before issuing a prompt.

path
The list of directories in which to search for commands. path is initialized from the environment variable PATH, which the C shell updates whenever path changes. A null word ('') specifies the current directory. The default is typically (/usr/bin .). One may override this initial search path upon csh start-up by setting it in .cshrc or .login (for login shells only). If path becomes unset, only full pathnames will execute. An interactive C shell will normally hash the contents of the directories listed after reading .cshrc, and whenever path is reset. If new commands are added, use the rehash command to update the table.

prompt
The string an interactive C shell prompts with. Noninteractive shells leave the prompt variable unset. Aliases and other commands in the .cshrc file that are only useful interactively, can be placed after the following test: 'if ($?prompt == 0) exit', to reduce startup time for noninteractive shells. A ! in the prompt string is replaced by the current event number. The default prompt is hostname% for mere mortals, or hostname# for the privileged user.

The setting of $prompt has three meanings:

$prompt not set non-interactive shell, test $?prompt.
$prompt set but == "" .cshrc called by the which(1) command.
$prompt set and != "" normal interactive shell.
savehist
The number of lines from the history list that are saved in
~/.history when the user logs out. Large values for savehist
slow down the C shell during startup.

shell
The file in which the C shell resides. This is used in forking shells
to interpret files that have execute bits set, but that are not
executable by the system.

status
The status returned by the most recent command. If that command
terminated abnormally, 0200 is added to the status. Built-in
commands that fail return exit status 1; all other built-in
commands set status to 0.

time
Control automatic timing of commands. Can be supplied with one
or two values. The first is the reporting threshold in CPU seconds.
The second is a string of tags and text indicating which resources
to report on. A tag is a percent sign (%) followed by a single
upper-case letter (unrecognized tags print as text):

  %D   Average amount of unshared data space used in
       Kilobytes.
  %E   Elapsed (wallclock) time for the command.
  %F   Page faults.
  %I   Number of block input operations.
  %K   Average amount of unshared stack space used in
       Kilobytes.
  %M   Maximum real memory used during execution of the
       process.
  %O   Number of block output operations.
  %P   Total CPU time — U (user) plus S (system) — as a
       percentage of E (elapsed) time.
  %S   Number of seconds of CPU time consumed by the
       kernel on behalf of the user’s process.
  %U   Number of seconds of CPU time devoted to the user’s
       process.
  %W   Number of swaps.
  %X   Average amount of shared memory used in Kilobytes.

The default summary display outputs from the %U, %S, %E, %P, %X,
%D, %I, %O, %F, and %W tags, in that order.

verbose
Display each command after history substitution takes place.
csh(1)

**Large File Behavior**
See `largefile(5)` for the description of the behavior of `csh` when encountering files greater than or equal to 2 Gbyte (2³¹ bytes).

**FILES**
- `~/.cshrc` Read at beginning of execution by each shell.
- `~/.login` Read by login shells after `.cshrc` at login.
- `~/.logout` Read by login shells at logout.
- `~/.history` Saved history for use at next login.
- `/usr/bin/sh` The Bourne shell, for shell scripts not starting with a ‘#’.
- `/tmp/sh*` Temporary file for ‘<<’.
- `/etc/passwd` Source of home directories for ‘~name’.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

**SEE ALSO**
`bc(1), echo(1), login(1), ls(1), more(1), ps(1), sh(1), shell_builtins(1),
tset(1B), which(1), df(1M), swap(1M), sysdef(1M), access(2), exec(2), fork(2),
pipe(2), a.out(4), environ(4), ascii(5), attributes(5), environ(5),
largefile(5), termio(7)`

**DIAGNOSTICS**
You have stopped jobs.
You attempted to exit the C shell with stopped jobs under job control. An immediate second attempt to exit will succeed, terminating the stopped jobs.

**WARNINGS**
The use of `setuid` shell scripts is strongly discouraged.

**NOTES**
Words can be no longer than 1024 bytes. The system limits argument lists to 1,048,576 bytes. However, the maximum number of arguments to a command for which filename expansion applies is 1706. Command substitutions may expand to no more characters than are allowed in the argument list. To detect looping, the shell restricts the number of alias substitutions on a single line to 20.

When a command is restarted from a stop, the shell prints the directory it started in if this is different from the current directory; this can be misleading (that is, wrong) as the job may have changed directories internally.

Shell built-in functions are not stoppable/restartable. Command sequences of the form `a ; b ; c` are also not handled gracefully when stopping is attempted. If you suspend `b`, the shell never executes `c`. This is especially noticeable if the expansion results from an alias. It can be avoided by placing the sequence in parentheses to force it into a subshell.
Control over terminal output after processes are started is primitive; use the Sun Window system if you need better output control.

Commands within loops, prompted for by ?, are not placed in the history list.

Control structures should be parsed rather than being recognized as built-in commands. This would allow control commands to be placed anywhere, to be combined with |, and to be used with & and ; metasyntax.

It should be possible to use the : modifiers on the output of command substitutions. There are two problems with : modifier usage on variable substitutions: not all of the modifiers are available, and only one modifier per substitution is allowed.

The g (global) flag in history substitutions applies only to the first match in each word, rather than all matches in all words. The common text editors consistently do the latter when given the g flag in a substitution command.

Quoting conventions are confusing. Overriding the escape character to force variable substitutions within double quotes is counterintuitive and inconsistent with the Bourne shell.

Symbolic links can fool the shell. Setting the hardpaths variable alleviates this.

It is up to the user to manually remove all duplicate pathnames accrued from using built-in commands as

```csh
set path = pathnames
```

```csh
setenv PATH = pathnames
```

more than once. These often occur because a shell script or a .cshrc file does something like

```csh
' set path=/usr/local /usr/hosts $path' to ensure that the named directories are in the pathname list.
```

The only way to direct the standard output and standard error separately is by invoking a subshell, as follows:

```csh
command > outfile ) >& errorfile
```

Although robust enough for general use, adventures into the esoteric periphery of the C shell may reveal unexpected quirks.

If you start csh as a login shell and you do not have a .login in your home directory, then the csh reads in the /etc/.login.

When the shell executes a shell script that attempts to execute a non-existent command interpreter, the shell returns an erroneous diagnostic message that the shell script file does not exist.

**BUGS**

As of this writing, the time built-in command does NOT compute the last 6 fields of output, rendering the output to erroneously report the value "0" for these fields:
example `time ls -R`
9.0u 11.0s 3:32 10% 0+0k 0+0io 0pf+0w
The `csplit` utility reads the file named by the `file` operand, writes all or part of that file into other files as directed by the `arg` operands, and writes the sizes of the files.

The following options are supported:

- `-f prefix` Name the created files `prefix00`, `prefix01`, ..., `prefixn`. The default is `xx00` ... `xxn`. If the `prefix` argument would create a file name exceeding 14 bytes, an error will result; `csplit` will exit with a diagnostic message and no files will be created.

- `-k` Leave previously created files intact. By default, `csplit` will remove created files if an error occurs.

- `-n number` Use `number` decimal digits to form filenames for the file pieces. The default is 2.

- `-s` Suppress the output of file size messages.

The following operands are supported:

- `file` The path name of a text file to be split. If `file` is `-`, the standard input will be used.

- `arg1 ... argn` can be a combination of the following:

  - `/rexp/[offset]` Create a file using the content of the lines from the current line up to, but not including, the line that results from the evaluation of the regular expression with `offset`, if any, applied. The regular expression `rexp` must follow the rules for basic regular expressions. The optional `offset` must be a positive or negative integer value representing a number of lines. The integer value must be preceded by `+` or `−`. If the selection of lines from an offset expression of this type would create a file with zero lines, or one with greater than the number of lines left in the input file, the results are unspecified. After the section is created, the current line will be set to the line that results from the evaluation of the regular expression with any `offset` applied. The pattern match of `rexp` always is applied from the current line to the end of the file.

  - `%%rexp%[offset]` This operand is the same as `/rexp/[offset]`, except that no file will be created for the selected section of the input file.

  - `line_no` Create a file from the current line up to (but not including) the line number `line_no`. Lines in the file will be numbered starting at one. The current line becomes `line_no`.

  - `{num}` Repeat operand. This operand can follow any of the operands described previously. If it follows a `rexp` type operand, that operand will be applied `num` more times. If it follows a `line_no`
operand, the file will be split every line_no lines, num times, from that point.

An error will be reported if an operand does not reference a line between the current position and the end of the file.

**USAGE**

See largefile(5) for the description of the behavior of csplit when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

**EXAMPLES**

**EXAMPLE 1** Splitting and combining files

This example creates four files, cobol00...cobol03.

```
example% csplit -f cobol filename '/procedure division/' /par5./ /par6./
```

After editing the “split” files, they can be recombined as follows:

```
example% cat cobol[0-3] > filename
```

Note: This example overwrites the original file.

**EXAMPLE 2** Splitting a file into equal parts

This example splits the file at every 100 lines, up to 10,000 lines. The -k option causes the created files to be retained if there are less than 10,000 lines; however, an error message would still be printed.

```
example% csplit -k filename 100 {99}
```

**EXAMPLE 3** Creating a file for separate C routines

If prog.c follows the normal C coding convention (the last line of a routine consists only of a } in the first character position), this example creates a file for each separate C routine (up to 21) in prog.c.

```
example% csplit -k prog.c 'main\% \"/\"}/+1' {20}
```

**ENVIRONMENT VARIABLES**

See environ(5) for descriptions of the following environment variables that affect the execution of csplit: LC_COLLATE, LC_CTYPE, LC_MESSAGES, and NLSPATH.

**EXIT STATUS**

The following exit values are returned:

- 0 Successful completion.
- >0 An error occurred.

**ATTRIBUTES**

See attributes(5) for descriptions of the following attributes:
The diagnostic messages are self-explanatory, except for the following:

\texttt{arg} – out of range

The given argument did not reference a line between the current position and the end of the file.

\begin{tabular}{|l|l|}
\hline
\textbf{ATTRIBUTE TYPE} & \textbf{ATTRIBUTE VALUE} \\
\hline
Availability & SUNWesu \\
CSI & Enabled \\
\hline
\end{tabular}

SEE ALSO sed(1), split(1), attributes(5), environ(5), largefile(5)

DIAGNOSTICS
NAME
ct – spawn login to a remote terminal

SYNOPSIS
cr [options] telno...

DESCRIPTION
The ct utility dials the telephone number of a modem that is attached to a terminal
and spawns a login process to that terminal. The telno is a telephone number, with
equal signs for secondary dial tones and minus signs for delays at appropriate places.
(The set of legal characters for telno is 0 through 9, -, =, *, and #. The maximum length
telno is 31 characters). If more than one telephone number is specified, ct will try each
in succession until one answers; this is useful for specifying alternate dialing paths.

ct will try each line listed in the file /etc/uucp/Devices until it finds an available
line with appropriate attributes, or runs out of entries.

After the user on the destination terminal logs out, there are two things that could
occur depending on what type of port monitor is monitoring the port. In the case of no
port monitor, ct prompts: Reconnect? If the response begins with the letter n, the
line will be dropped; otherwise, ttymon will be started again and the login: prompt
will be printed. In the second case, where a port monitor is monitoring the port, the
port monitor reissues the login: prompt.

The user should log out properly before disconnecting.

OPTIONS
The following options are supported:

-h
Normally, ct will hang up the current line so that it can be used to
answer the incoming call. The -h option will prevent this action.
The -h option will also wait for the termination of the specified ct
process before returning control to the user’s terminal.

-s speed
The data rate may be set with the -s option. speed is expressed in
baud rates. The default baud rate is 1200.

-v
If the -v (verbose) option is used, ct will send a running narrative
to the standard error output stream.

-w n
If there are no free lines ct will ask if it should wait, and for how
many minutes, before it gives up. ct will continue to try to open
the dialers at one-minute intervals until the specified limit is
exceeded. This dialogue may be overridden by specifying the -wn
option. n is the maximum number of minutes that ct is to wait for
a line.

-x n
This option is used for debugging; it produces a detailed output of
the program execution on stderr. n is a single number between 0
and 9. As n increases to 9, more detailed debugging information is
given.
FILES
/etc/uucp/Devices
/var/adm/ctlog

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWbnuu</td>
</tr>
</tbody>
</table>

SEE ALSO
cu(1C), login(1), uucp(1C), ttymon(1M), attributes(5)

NOTES
The ct program will not work with a DATAKIT Multiplex interface.

For a shared port, one used for both dial-in and dial-out, the ttymon program running on the line must have the -r and -b options specified (see ttymon(1M)).
### NAME

ctags – create a tags file for use with ex and vi

### SYNOPSIS

```
/usr/bin/ctags [\-aBFuvwx] [\-f tagsfile] file...
```

```
/usr/xpg4/bin/ctags [\-aBFuvwx] [\-f tagsfile] file...
```

### DESCRIPTION

The `ctags` utility makes a tags file for `ex(1)` from the specified C, C++, Pascal, FORTRAN, `yacc(1)`, and `lex(1)` sources. A tags file gives the locations of specified objects (in this case functions and typedefs) in a group of files. Each line of the tags file contains the object name, the file in which it is defined, and an address specification for the object definition. Functions are searched with a pattern, typedefs with a line number. Specifiers are given in separate fields on the line, separated by SPACE or TAB characters. Using the tags file, `ex` can quickly find these objects’ definitions.

Normally, `ctags` places the tag descriptions in a file called `tags`; this may be overridden with the `-f` option.

Files with names ending in `.c` or `.h` are assumed to be either C or C++ source files and are searched for C/C++ routine and macro definitions. Files with names ending in `.cc`, `.C`, or `.cxx`, are assumed to be C++ source files. Files with names ending in `.y` are assumed to be yacc source files. Files with names ending in `.i` are assumed to be lex files. Others are first examined to see if they contain any Pascal or FORTRAN routine definitions; if not, they are processed again looking for C definitions.

The tag `main` is treated specially in C or C++ programs. The tag formed is created by prepending `M` to `file`, with a trailing `.c`, `.cc`, `.C`, or `.cxx` removed, if any, and leading path name components also removed. This makes use of `ctags` practical in directories with more than one program.

### OPTIONS

The precedence of the options that pertain to printing is `-x`, `-v`, then the remaining options. The following options are supported:

- `-a`        Appends output to an existing `tags` file.
- `-B`        Uses backward searching patterns (\?...\?).
- `-f tagsfile` Places the tag descriptions in a file called `tagsfile` instead of `tags`.
- `-F`        Uses forward searching patterns (/.../) (default).
- `-t`        Creates tags for typedefs. `/usr/xpg4/bin/ctags` creates tags for typedefs by default.
- `-u`        Updates the specified files in tags, that is, all references to them are deleted, and the new values are appended to the file. Beware: this option is implemented in a way that is rather slow; it is usually faster to simply rebuild the `tags` file.
- `-v`        Produces on the standard output an index listing the function name, file name, and page number (assuming 64 line pages). Since the output will be sorted into lexicographic order, it may be desired to run the output through `sort -f`.

---

252 man pages section 1: User Commands • Last Revised 18 Mar 1997
ctags(1)

- \( w \)
  Suppresses warning diagnostics.

- \( x \)
  Produces a list of object names, the line number and file name on
  which each is defined, as well as the text of that line and prints this
  on the standard output. This is a simple index which can be
  printed out as an off-line readable function index.

**OPERANDS**

The following `file` operands are supported:

- `file.c`
  Files with basenames ending with the `.c` suffix are treated as
  C-language source code.

- `file.h`
  Files with basenames ending with the `.h` suffix are treated as
  C-language source code.

- `file.f`
  Files with basenames ending with the `.f` suffix are treated as
  FORTRAN-language source code.

**USAGE**

The `-v` option is mainly used with `vgrind` which will be part of the optional BSD
Compatibility Package.

**EXAMPLES**

**EXAMPLE 1** Producing entries in alphabetical order

Using `ctags` with the `-v` option produces entries in an order which may not always
be appropriate for `vgrind`. To produce results in alphabetical order, you may want to
run the output through `sort -f`.

```
example% ctags -v filename.c filename.h | sort -f > index
example% vgrind -x index
```

**EXAMPLE 2** Building a tags file

To build a tags file for C sources in a directory hierarchy rooted at `sourcedir`, first create
an empty tags file, and then run `find(1)`

```
example% cd sourcedir ; rm -f tags ; touch tags
example% find . \( -name SCCS -prune -name \`
  \'*.c' -o -name \'*.h' \) -exec ctags -u {} \;
```

Notice that spaces must be entered exactly as shown.

**ENVIRONMENT VARIABLES**

See `environ(5)` for descriptions of the following environment variables that affect the
execution of `ctags`: `LC_COLLATE`, `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

**EXIT STATUS**

The following exit values are returned:

- `0`
  Successful completion.

- `>0`
  An error occurred.

**FILES**

- `tags`
  output tags file

**ATTRIBUTES**

See `attributes(5)` for descriptions of the following attributes:
Recognition of functions, subroutines, and procedures for FORTRAN and Pascal is done in a very simpleminded way. No attempt is made to deal with block structure; if you have two Pascal procedures in different blocks with the same name, you lose.

The method of deciding whether to look for C or Pascal and FORTRAN functions is a hack.

The `ctags` utility does not know about `#ifdefs`.

The `ctags` utility should know about Pascal types. Relies on the input being well formed to detect typedefs. Use of `-tx` shows only the last line of typedefs.
cu(1C)

NAME

cu – call another UNIX system

SYNOPSIS


DESCRIPTION

The command cu calls up another UNIX system, a terminal, or possibly a non-UNIX system. It manages an interactive conversation with possible transfers of files. It is convenient to think of cu as operating in two phases. The first phase is the connection phase in which the connection is established. cu then enters the conversation phase. The -d option is the only one that applies to both phases.

OPTIONS

cu accepts many options. The -c, -l, and -s options play a part in selecting the medium. The remaining options are used in configuring the line.

- b bits

Forces bits to be the number of bits processed on the line. bits is either 7 or 8. This allows connection between systems with different character sizes. By default, the character size of the line is set to the same value as the current local terminal, but the character size setting is affected by LC_CTYPE also.

- c device

Forces cu to use only entries in the "Type" field (the first field in the /etc/uucp/Devices file) that match the user specified device, usually the name of a local area network.

- C

Runs the local-cmd specified at the end of the command line instead of entering interactive mode. The stdin and stdout of the command that is run refer to the remote connection.

- d

Prints diagnostic traces.

- e

Sets an EVEN data parity. This option designates that EVEN parity is to be generated for data sent to the remote system.

- h

Sets communication mode to half-duplex. This option emulates local echo in order to support calls to other computer systems that expect terminals to be set to half-duplex mode.

- H

Ignores one hangup. This allows the user to remain in cu while the remote machine disconnects and places a call back to the local machine. This option should be used when connecting to systems with callback or dialback modems. Once the callback occurs subsequent hangups will cause cu to terminate. This option can be specified more than once. For more information about dialback configuration, see remote(4) and System Administration Guide: IP Services

- l line

Specifies a device name to use as the communication line. This can be used to override the search that would otherwise take place for the first available line having the right speed. When the -l option is used without the -s option, the speed of a line is taken from the /etc/uucp/Devices file record in which line matches the second field (the Line field). When the -l and -s options are both used
cu(1C) together, cu will search the /etc/uucp/Devices file to check if the requested speed for the requested line is available. If so, the connection will be made at the requested speed, otherwise, an error message will be printed and the call will not be made. In the general case where a specified device is a directly connected asynchronous line (for instance, /dev/term/a), a telephone number (telno) is not required. The specified device need not be in the /dev directory. If the specified device is associated with an auto dialer, a telephone number must be provided.

-\L\ Goes through the login chat sequence specified in the /etc/uucp/Systems file. For more information about the chat sequence, see System Administration Guide: IP Services

-\n\ Requests user prompt for telephone number. For added security, this option will prompt the user to provide the telephone number to be dialed, rather than taking it from the command line.

-\o\ Sets an ODD data parity. This option designates that ODD parity is to be generated for data sent to the remote system.

-\s\ speed\ Specifies the transmission speed (300, 1200, 2400, 4800, 9600, 19200, 38400). The default value is "Any" speed which will depend on the order of the lines in the /etc/uucp/Devices file.

-\t\ Dials a terminal which has been set to auto answer. Appropriate mapping of carriage-return to carriage-return-line-feed pairs is set.

OPERANDS The following operands are supported:

telno\ When using an automatic dialler, specifies the telephone number with equal signs for secondary dial tone or minus signs placed appropriately for delays of 4 seconds.

systemname\ Specifies a uucp system name, which can be used rather than a telephone number; in this case, cu will obtain an appropriate direct line or telephone number from a system file.

Connection Phase cu uses the same mechanism that uucp(1C) does to establish a connection. This means that it will use the uucp control files /etc/uucp/Devices and /etc/uucp/Systems. This gives cu the ability to choose from several different media to establish the connection. The possible media include telephone lines, direct connections, and local area networks (LAN). The /etc/uucp/Devices file contains a list of media that are available on your system. The /etc/uucp/Systems file contains information for connecting to remote systems, but it is not generally readable.

Note: cu determines which /etc/uucp/Systems and /etc/uucp/Devices files to use based upon the name used to invoke cu. In the simple case, this name will be "cu", but you could also have created a link to cu with another name, such as "pppcu", in which case cu would then look for a "service=pppcu" entry in the /etc/uucp/Sysfiles file to determine which /etc/uucp/Systems file to use.
The `telno` or `systemname` parameter from the command line is used to tell `cu` what system you wish to connect to. This parameter can be blank, a telephone number, a system name, or a LAN specific address.

**telephone number**
A telephone number is a string consisting of the tone dial characters (the digits 0 through 9, *, and #) plus the special characters = and −. The equal sign designates a secondary dial tone and the minus sign creates a 4 second delay.

**system name**
A system name is the name of any computer that `uucp` can call; the `uname(1C)` command prints a list of these names.

**LAN address**
The documentation for your LAN will show the form of the LAN specific address.

If `cu`'s default behavior is invoked (not using the `-c` or `-l` options), `cu` will use the `telno` or `systemname` parameter to determine which medium to use. If a telephone number is specified, `cu` will assume that you wish to use a telephone line and it will select an automatic call unit (ACU). Otherwise, `cu` will assume that it is a system name. `cu` will follow the `uucp` calling mechanism and use the `/etc/uucp/Systems` and `/etc/uucp/Devices` files to obtain the best available connection. Since `cu` will choose a speed that is appropriate for the medium that it selects, you may not use the `-s` option when this parameter is a system name.

The `-c` and `-l` options modify this default behavior. `-c` is most often used to select a LAN by specifying a Type field from the `/etc/uucp/Devices` file. You must include either a `telno` or `systemname` value when using the `-c` option. If the connection to `systemname` fails, a connection will be attempted using `systemname` as a LAN specific address. The `-l` option is used to specify a device associated with a direct connection. If the connection is truly a direct connection to the remote machine, then there is no need to specify a `systemname`. This is the only case where a `telno` or `systemname` parameter is unnecessary. On the other hand, there may be cases in which the specified device connects to a dialer, so it is valid to specify a telephone number. The `-c` and `-l` options should not be specified on the same command line.

**Conversation Phase**
After making the connection, `cu` runs as two processes. The `transmit` process reads data from the standard input and, except for lines beginning with ~, passes it to the remote system. The `receive` process accepts data from the remote system and, except for lines beginning with ~, passes it to the standard output. Normally, an automatic DC3/DC1 protocol is used to control input from the remote so the buffer is not overrun. Lines beginning with ~ have special meanings.

**Commands**
The `transmit` process interprets the following user initiated commands:

- `.` Terminates the conversation.
- `!` Escapes to an interactive shell on the local system.
- `!cmd` Runs `cmd` on the local system (via `sh -c`).
cu(1C)

-\$cmd... Runs \texttt{cmd} locally and send its output to the remote system.

-\%cd Changes the directory on the local system. Note: \texttt{-l\%cd} will cause the command to be run by a sub-shell, probably not what was intended.

-\%take from [ to ] Copies file \texttt{from} (on the remote system) to file \texttt{to} on the local system. If \texttt{to} is omitted, the \texttt{from} argument is used in both places.

-\%put from [ to ] Copies file \texttt{from} (on local system) to file \texttt{to} on remote system. If \texttt{to} is omitted, the \texttt{from} argument is used in both places.

-- line Sends the line \texttt{-- line} to the remote system.

-\%break Transmits a \texttt{BREAK} to the remote system (which can also be specified as \texttt{--\%b}).

-\%debug Toggles the \texttt{-d} debugging option on or off (which can also be specified as \texttt{--\%d}).

-t Prints the values of the termio structure variables for the user's terminal (useful for debugging).

-l Prints the values of the termio structure variables for the remote communication line (useful for debugging).

-\%ifc Toggles between DC3/DC1 input control protocol and no input control. This is useful when the remote system does not respond properly to the DC3 and DC1 characters (can also be specified as \texttt{--noostop}).

-\%ofc Toggles the output flow control setting. When enabled, outgoing data may be flow controlled by the remote host (can also be specified as \texttt{--noostop}).

-\%divert Allows/disallows unsolicited diversions. That is, diversions not specified by \texttt{--take}.

-\%old Allows/disallows old style syntax for received diversions.

-\%nostop Same as \texttt{--ifc}.

The receive process normally copies data from the remote system to the standard output of the local system. It may also direct the output to local files.

The use of \texttt{--put} requires \texttt{stty(1)} and \texttt{cat(1)} on the remote side. It also requires that the current erase and kill characters on the remote system be identical to these current control characters on the local system. Backslashes are inserted at appropriate places.
The use of `~` requires the existence of `echo(1)` and `cat(1)` on the remote system, and that the remote system must be using the Bourne shell, `sh`. Also, tabs mode (see `stty(1)`) should be set on the remote system if tabs are to be copied without expansion to spaces.

When `cu` is used on system X to connect to system Y and subsequently used on system Y to connect to system Z, commands on system Y can be executed by using `~`. Executing a tilde command reminds the user of the local system `uname`. For example, `uname` can be executed on Z, X, and Y as follows:

```
uname
Z
~[X]!uname
X
~[Y]!uname
Y
```

In general, `~` causes the command to be executed on the original machine. `~~` causes the command to be executed on the next machine in the chain.

**EXAMPLES**

**EXAMPLE 1** Dialling a system

To dial a system whose telephone number is 9 1 201 555 1234 using 1200 baud (where dialtone is expected after the 9):

```
example% cu -s 1200 9=12015551234
```

If the speed is not specified, “Any” is the default value.

**EXAMPLE 2** Logging in to a system on a direct line

To login to a system connected by a direct line:

```
example% cu -l /dev/term/b
```

or

```
example% cu -l term/b
```

**EXAMPLE 3** Dialling a system with specific line and speed

To dial a system with a specific line and speed:

```
example% cu -s 1200 -l term/b
```

**EXAMPLE 4** Using a system name

To use a system name:

```
example% cu systemname
```

**ENVIRONMENT VARIABLES**

See `environ(5)` for descriptions of the following environment variables that affect the execution of `cu`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`. 
cu(1C)

EXIT STATUS  The following exit values are returned:
0       Successful completion.
>0       An error occurred.

FILES
/etc/uucp/Devices   device file
/etc/uucp/Sysfiles  system file
/etc/uucp/Systems   system file
/var/spool/locks/*  lock file

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWbnuu</td>
</tr>
</tbody>
</table>

SEE ALSO  cat(1), echo(1), stty(1), tip(1), uname(1), ct(1C), uuname(1C), uucp(1C), remote(4), attributes(5), environ(5)

System Administration Guide: IP Services

NOTES  The cu utility takes the default action upon receipt of signals, with the exception of:

SIGHUP     Close the connection and terminate.
SIGINT     Forward to the remote system.
SIGQUIT    Forward to the remote system.
SIGUSR1    Terminate the cu process without the normal connection closing sequence.

The cu command does not do any integrity checking on data it transfers. Data fields with special cu characters may not be transmitted properly. Depending on the interconnection hardware, it may be necessary to use a -. to terminate the conversion, even if stty 0 has been used. Non-printing characters are not dependably transmitted using either the -%put or -%take commands. -%put and -%take cannot be used over multiple links. Files must be moved one link at a time.

There is an artificial slowing of transmission by cu during the -%put operation so that loss of data is unlikely. Files transferred using -%take or -%put must contain a trailing newline, otherwise, the operation will hang. Entering a Control-D command usually clears the hang condition.
**NAME**  
cut – cut out selected fields of each line of a file

**SYNOPSIS**  
cut -b list [-n] [file...]

cut -c list [file...]

cut -f list [-a delim] [-s] [file...]

**DESCRIPTION**  
Use the cut utility to cut out columns from a table or fields from each line of a file; in data base parlance, it implements the projection of a relation. The fields as specified by list can be fixed length, that is, character positions as on a punched card (-c option) or the length can vary from line to line and be marked with a field delimiter character like TAB (-f option). cut can be used as a filter.

Either the -b, -c, or -f option must be specified.

Use grep(1) to make horizontal “cuts” (by context) through a file, or paste(1) to put files together column-wise (that is, horizontally). To reorder columns in a table, use cut and paste.

**OPTIONS**  
The following options are supported:

- **list**  
A comma-separated or blank-character-separated list of integer field numbers (in increasing order), with optional – to indicate ranges (for instance, 1, 4, 7; 1–3, 8; –5, 10 (short for 1–5, 10); or 3– (short for third through last field)).

- **-b list**  
The list following -b specifies byte positions (for instance, -b1,72 would pass the first 72 bytes of each line). When -b and -n are used together, list is adjusted so that no multi-byte character is split.

- **-c list**  
The list following -c specifies character positions (for instance, -c1,72 would pass the first 72 characters of each line).

- **-a delim**  
The character following -a is the field delimiter (-f option only). Default is tab. Space or other characters with special meaning to the shell must be quoted. delim can be a multi-byte character.

- **-f list**  
The list following -f is a list of fields assumed to be separated in the file by a delimiter character (see -a); for instance, -f1,7 copies the first and seventh field only. Lines with no field delimiters will be passed through intact (useful for table subheadings), unless -s is specified.

- **-n**  
Do not split characters. When -b list and -n are used together, list is adjusted so that no multi-byte character is split.

- **-s**  
Suppresses lines with no delimiter characters in case of -f option. Unless specified, lines with no delimiters will be passed through untouched.

**OPERANDS**  
The following operands are supported:
file

A path name of an input file. If no file operands are specified, or if a file operand is --, the standard input will be used.

**USAGE**

See largefile(5) for the description of the behavior of cut when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

**EXAMPLES**

**EXAMPLE 1** Mapping user IDs

A mapping of user IDs to names follows:

```bash
example$ cut -d: -f1,5 /etc/passwd
```

**EXAMPLE 2** Setting current login name

To set name to current login name:

```bash
example$ name=`who am i | cut -f1 -d ''`
```

**ENVIRONMENT VARIABLES**

See environ(5) for descriptions of the following environment variables that affect the execution of cut: LC_CTYPE, LC_MESSAGES, and NLSPATH.

**EXIT STATUS**

The following exit values are returned:

0 All input files were output successfully.

>0 An error occurred.

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

**SEE ALSO**

grep(1), paste(1), attributes(5), environ(5), largefile(5)

**DIAGNOSTICS**

cut: -n may only be used with -b

cut: -d may only be used with -f

cut: -s may only be used with -f

cut: cannot open <file>

Either file cannot be read or does not exist. If multiple files are present, processing continues.

cut: no delimiter specified

Missing delim on -d option.
cut: invalid delimiter

cut: no list specified
   Missing list on -b, -c, or -f option.

cut: invalid range specifier

cut: too many ranges specified

cut: range must be increasing

cut: invalid character in range

cut: internal error processing input

cut: invalid multibyte character

cut: unable to allocate enough memory
date(1)

NAME          date – write the date and time

SYNOPSIS     /usr/bin/date [-u] [+ format]
              /usr/bin/date [-a [-] sss.fff]
              /usr/bin/date [-u] [ [mmdd] HHMM | mmdHHMM [cc] yy] [.SS]
              /usr/xpg4/bin/date [-u] [+ format]
              /usr/xpg4/bin/date [-a [-] sss.fff]
              /usr/xpg4/bin/date [-u] [ [mmdd] HHMM | mmdHHMM [cc] yy] [.SS]

DESCRIPTION  The date utility writes the date and time to standard output or attempts to set
              the system date and time. By default, the current date and time will be written.

Specifications of native language translations of month and weekday names are
supported. The month and weekday names used for a language are based on the
locale specified by the environment variable LC_TIME; see environ(5).

The following is the default form for the "C" locale:

%a %b %e %T %Z %Y
for example,
Fri Dec 23 10:10:42 EST 1988

OPTIONS       The following options are supported:

   -a [-] sss.fff       Slowly adjust the time by sss fff seconds (fff represents fractions of a
                        second). This adjustment can be positive or negative. The system’s
                        clock will be sped up or slowed down until it has drifted by the
                        number of seconds specified. Only the super-user may adjust the
                        time.

   -u                   Display (or set) the date in Greenwich Mean Time
                        (GMT—universal time), bypassing the normal conversion to (or
                        from) local time.

OPERANDS      The following operands are supported:

   +format             If the argument begins with +, the output of date is the result of
                        passing format and the current time to strftime(). date uses
                        the conversion specifications listed on the strftime(3C) manual
                        page, with the conversion specification for %C determined by
                        whether /usr/bin/date or /usr/xpg4/bin/date is used:
                        /usr/bin/date
                        Locale’s date and time
                        representation. This is the default output for date.

                        /usr/xpg4/bin/date
                        Century (a year divided by 100 and
                        truncated to an integer) as a
                        decimal number [00-99].

264  man pages section 1: User Commands • Last Revised 12 Dec 2000
The string is always terminated with a NEWLINE. An argument containing blanks must be quoted; see the EXAMPLES section.

- **mm**: Month number
- **dd**: Day number in the month
- **HH**: Hour number (24 hour system)
- **MM**: Minute number
- **SS**: Second number
- **cc**: Century (a year divided by 100 and truncated to an integer) as a decimal number [00-99]. For example, cc is 19 for the year 1988 and 20 for the year 2007.
- **yy**: Last two digits of the year number. If century (cc) is not specified, then values in the range 69–99 shall refer to years 1969 to 1999 inclusive, and values in the range 00–68 shall refer to years 2000 to 2068, inclusive.

The month, day, year number, and century may be omitted; the current values are applied as defaults. For example, the following entry:

```
example$ date 10080045
```

sets the date to Oct 8, 12:45 a.m. The current year is the default because no year is supplied. The system operates in GMT. `date` takes care of the conversion to and from local standard and daylight time. Only the super-user may change the date. After successfully setting the date and time, `date` displays the new date according to the default format. The `date` command uses `TZ` to determine the correct time zone information; see `environ(5)`.

### EXAMPLES

#### EXAMPLE 1 Generating output

The command

```
example$ date '+DATE: %m/%d/%Y
TIME: %H:%M:%S'
```

generates as output

```
DATE: 08/01/76
TIME: 14:45:05
```

#### EXAMPLE 2 Setting the current time

The command

```
example$ date 1234.56
```

sets the current time to **12:34:56**.
EXAMPLE 3 Setting another time and date in Greenwich Mean Time

The command

```
example# date -u 010100302000
```

sets the date to January 1st, 12:30 am, 2000, which will be displayed as

```
Thu Jan 01 00:30:00 GMT 2000
```

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of date: LC_CTYPE, LC_TIME, LC_MESSAGES, and NLSPATH.

TZ  Determine the timezone in which the time and date are written, unless the -u option is specified. If the TZ variable is not set and the -u is not specified, the system default timezone is used.

EXIT STATUS

The following exit values are returned:

0  Successful completion.

>0  An error occurred.

ATTRIBUTES

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

```
ATTRIBUTE TYPE     ATTRIBUTE VALUE
/usr/bin/date
Availability        SUNWcsu
CSI                enabled

/usr/xpg4/bin/date
Availability        SUNWxcu4
CSI                enabled
```

SEE ALSO

strftime(3C), attributes(5), environ(5), XPG4(5)

DIAGNOSTICS

no permission  You are not the super-user and you tried to change the date.

bad conversion  The date set is syntactically incorrect.

NOTES

If you attempt to set the current date to one of the dates that the standard and alternate time zones change (for example, the date that daylight time is starting or ending), and you attempt to set the time to a time in the interval between the end of standard time and the beginning of the alternate time (or the end of the alternate time and the beginning of standard time), the results are unpredictable.
Using the `date` command from within windowing environments to change the date can lead to unpredictable results and is unsafe. It may also be unsafe in the multi-user mode, that is, outside of a windowing system, if the date is changed rapidly back and forth. The recommended method of changing the date is `date -a`.
dc is an arbitrary precision arithmetic package. Ordinarily it operates on decimal integers, but one may specify an input base, output base, and a number of fractional digits to be maintained. The overall structure of dc is a stacking (reverse Polish) calculator. If an argument is given, input is taken from that file until its end, then from the standard input.

bc is a preprocessor for dc that provides infix notation and a C-like syntax that implements functions. bc also provides reasonable control structures for programs. See bc(1).

The following constructions are recognized:

- **Number**: The value of the number is pushed on the stack. A number is an unbroken string of the digits 0–9. It may be preceded by an underscore (_) to input a negative number. Numbers may contain decimal points.
- **+ − / ∗ % ^**: The top two values on the stack are added (+), subtracted (−), multiplied (∗), divided (/), remaindered (%), or exponentiated (^). The two entries are popped off the stack; the result is pushed on the stack in their place. Any fractional part of an exponent is ignored.
- **sx**: The top of the stack is popped and stored into a register named x, where x may be any character. If the s is capitalized, x is treated as a stack and the value is pushed on it.
- **lx**: The value in register x is pushed on the stack. The register x is not altered. All registers start with zero value. If the l is capitalized, register x is treated as a stack and its top value is popped onto the main stack.
- **d**: The top value on the stack is duplicated.
- **p**: The top value on the stack is printed. The top value remains unchanged.
- **P**: Interprets the top of the stack as an ASCII string, removes it, and prints it.
- **f**: All values on the stack are printed.
Exits the program. If executing a string, the recursion level is popped by two.

Exits the program. The top value on the stack is popped and the string execution level is popped by that value.

Treats the top element of the stack as a character string and executes it as a string of dc commands.

Replaces the number on the top of the stack with its scale factor.

Puts the bracketed ASCII string onto the top of the stack.

The top two elements of the stack are popped and compared. Register \( x \) is evaluated if they obey the stated relation.

Replaces the top element on the stack by its square root. Any existing fractional part of the argument is taken into account, but otherwise the scale factor is ignored.

Interprets the rest of the line as a shell command.

All values on the stack are popped.

The top value on the stack is popped and used as the number radix for further input.

Pushes the input base on the top of the stack.

The top value on the stack is popped and used as the number radix for further output.

Pushes the output base on the top of the stack.

The top of the stack is popped, and that value is used as a non-negative scale factor: the appropriate number of places are printed on output, and maintained during multiplication, division, and exponentiation. The interaction of scale factor, input base, and output base will be reasonable if all are changed together.
dc(1)

K
Pushes the current scale factor on the top of the stack.

z
The stack level is pushed onto the stack.

Z
Replaces the number on the top of the stack with its length.

?
A line of input is taken from the input source (usually the terminal) and executed.

Y
Displays dc debugging information.

; :  are used by bc(1) for array operations.

EXAMPLES
EXAMPLE 1 Printing the first ten values of n!
This example prints the first ten values of n!:

```bash
[lal+dsa*pla10>y]sy
0sa1
lyx
```

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
</tbody>
</table>

SEE ALSO  bc(1), attributes(5)

DIAGNOSTICS
x is unimplemented  x is an octal number.
out of space        The free list is exhausted (too many digits).
out of stack space  Too many pushes onto the stack (stack overflow).
empty stack         Too many pops from the stack (stack underflow).
nesting depth       Too many levels of nested execution.
divide by 0          Division by zero.
sqrt of neg number  Square root of a negative number is not defined (no imaginary numbers).
exp not an integer   dc only processes integer exponentiation.
exp too big          The largest exponent allowed is 999.
input base is too large  The input base x: 2<= x <= 16.
<table>
<thead>
<tr>
<th>input base is too small</th>
<th>The input base x: 2&lt;= x &lt;= 16.</th>
</tr>
</thead>
<tbody>
<tr>
<td>output base is too large</td>
<td>The output base must be no larger than</td>
</tr>
<tr>
<td></td>
<td>BC_BASE_MAX.</td>
</tr>
<tr>
<td>invalid scale factor</td>
<td>Scale factor cannot be less than 1.</td>
</tr>
<tr>
<td>scale factor is too large</td>
<td>A scale factor cannot be larger than</td>
</tr>
<tr>
<td></td>
<td>BC_SCALE_MAX.</td>
</tr>
<tr>
<td>symbol table overflow</td>
<td>Too many variables have been specified.</td>
</tr>
<tr>
<td>invalid index</td>
<td>Index cannot be less than 1.</td>
</tr>
<tr>
<td>index is too large</td>
<td>An index cannot be larger than</td>
</tr>
<tr>
<td></td>
<td>BC_DIM_MAX.</td>
</tr>
</tbody>
</table>
The `deallocate` utility deallocates a device allocated to the evoking user. `device` can be a device defined in `device_allocate(4)` or one of the device special files associated with the device. It resets the ownership and the permission on all device special files associated with `device`, disabling the user’s access to that device. This option can be used by an authorized user to remove access to the device by another user. The required authorization is `solaris.device.allocate`.

When deallocation or forced deallocation is performed, the appropriate device cleaning program is executed, based on the contents of `device_allocate(4)`. These cleaning programs are normally stored in `/etc/security/lib`.

The following options are supported:

- `device` - Deallocate the device associated with the device special file specified by `device`.
- `-s` - Silent. Suppresses any diagnostic output.
- `-F device` - Forces deallocation of the device associated with the file specified by `device`. Only a user with the `solaris.devices.revoke` authorization is permitted to use this option.
- `-I` - Forces deallocation of all allocatable devices. Only a user with the `solaris.devices.revoke` authorization is permitted to use this option. This option should only be used at system initialization.

The following exit values are returned:

- non-zero - An error occurred.

The following files are used:

- `/etc/security/device_allocate`
- `/etc/security/device_maps`
- `/etc/security/dev/*`
- `/etc/security/lib/*`

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>
allocate(1), list_devices(1), bsmconv(1M), dminfo(1M),
device_allocate(4), device_maps(4), attributes(5)

SEE ALSO

NOTES The functionality described in this man page is available only if the Basic Security Module (BSM) has been enabled. See bsmconv(1M) for more information.
deroff(1)

NAME  
deroff – remove nroff/troff, tbl, and eqn constructs

SYNOPSIS  
deroff [-m [m | s | l]] [-w] [-i] [filename...]

DESCRIPTION  
deroff reads each of the filenames in sequence and removes all troff(1) requests, macro calls, backslash constructs, eqn(1) constructs (between .EQ and .EN lines, and between delimiters), and tbl(1) descriptions, perhaps replacing them with white space (blanks and blank lines), and writes the remainder of the file on the standard output. deroff follows chains of included files (.so and .nx troff commands); if a file has already been included, a .so naming that file is ignored and a .nx naming that file terminates execution. If no input file is given, deroff reads the standard input.

OPTIONS  
- m   The -m option may be followed by an m, s, or l. The -mm option causes the macros to be interpreted so that only running text is output (that is, no text from macro lines.) The -ml option forces the -mm option and also causes deletion of lists associated with the mm macros.

- w   If the -w option is given, the output is a word list, one “word” per line, with all other characters deleted. Otherwise, the output follows the original, with the deletions mentioned above. In text, a “word” is any string that contains at least two letters and is composed of letters, digits, ampersands (&), and apostrophes (‘); in a macro call, however, a “word” is a string that begins with at least two letters and contains a total of at least three letters. Delimiters are any characters other than letters, digits, apostrophes, and ampersands. Trailing apostrophes and ampersands are removed from “words.”

- i   The -i option causes deroff to ignore .so and .nx commands.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWdoc</td>
</tr>
</tbody>
</table>

SEE ALSO  
eqn(1), nroff(1), tbl(1), troff(1), attributes(5)

NOTES  
deroff is not a complete troff interpreter, so it can be confused by subtle constructs. Most such errors result in too much rather than too little output.

The -ml option does not handle nested lists correctly.
df(1B)

NAME
df – display status of disk space on file systems

SYNOPSIS
/usr/ucb/df [-a] [-i] [-t type] [filesystem...] [filename...]

DESCRIPTION
The df utility displays the amount of disk space occupied by currently mounted file systems, the amount of used and available space, and how much of the file system’s total capacity has been used.

If arguments to df are path names, df produces a report on the file system containing the named file. Thus ‘df .’ shows the amount of space on the file system containing the current directory.

OPTIONS
The following options are supported:

-a 
Report on all filesystems including the uninteresting ones which have zero total blocks (that is, auto-mounter).

-i 
Report the number of used and free inodes. Print ‘*’ if no information is available.

-t type
Report on filesystems of a given type (for example, nfs or ufs).

EXAMPLES
EXAMPLE 1 Output sample
A sample of output for df looks like:

eample$ df
Filesystem  kbytes used avail capacity Mounted on
sparky:/  7445  4714  1986  70% /  
sparky:/usr  42277  35291  2758  93% /usr

Note that used+avail is less than the amount of space in the file system (kbytes); this is because the system reserves a fraction of the space in the file system to allow its file system allocation routines to work well. The amount reserved is typically about 10%; this may be adjusted using tunefs (see tunefs(1M)). When all the space on a file system except for this reserve is in use, only the super-user can allocate new files and data blocks to existing files. When a file system is overallocated in this way, df may report that the file system is more than 100% utilized.

FILES
/etc/mnttab list of file systems currently mounted
/etc/vfstab list of default parameters for each file system

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

SEE ALSO
du(1M), quot(1M), tunefs(1M), mnttab(4), attributes(5)
The dhcpinfo utility prints the DHCP–supplied value(s) of the parameter requested on the command line. The parameter may be identified either by its numeric code in the DHCP specification, or by its mnemonic identifier, as listed in dhcp_inittab(4). This command is intended to be used in command substitutions in the shell scripts invoked by init(1M) at system boot. It first contacts the DHCP client daemon dhcpagent(1M) to verify that DHCP has successfully completed on the requested interface. If DHCP has successfully completed on the requested interface, dhcpinfo retrieves the values for the requested parameter. Parameter values echoed by dhcpinfo should not be used without checking its exit status. See EXIT STATUS.

See dhcp_inittab(4) for the list of mnemonic identifier codes for all DHCP parameters. See RFC 2132, DHCP Options and BOOTP Vendor Extensions for more detail.

**Output Format**

The output from dhcpinfo consists of one or more lines of ASCII text; the format of the output depends upon the requested parameter. The number of values returned per line and the total number of lines output for a given parameter are determined by the parameter’s granularity and maximum values, respectively, as defined by dhcp_inittab(4).

The format of each individual value is determined by the data type of the option, as determined by dhcp_inittab(4). The possible data types and their formats are listed below:

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Format</th>
<th>dhcp_inittab(4) type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unsigned Number</td>
<td>One or more decimal digits</td>
<td>UNUMBER8, UNUMBER16, UNUMBER32, UNUMBER64</td>
</tr>
<tr>
<td>Signed Number</td>
<td>One or more decimal digits, optionally preceded by a minus sign</td>
<td>SNUMBER8, SNUMBER16, SNUMBER32, SNUMBER64</td>
</tr>
<tr>
<td>IP Address</td>
<td>Dotted-decimal notation</td>
<td>IP</td>
</tr>
<tr>
<td>Octet</td>
<td>The string &quot;0x&quot; followed by a two-digit hexadecimal value</td>
<td>OCTET</td>
</tr>
<tr>
<td>String</td>
<td>Zero or more ASCII characters</td>
<td>ASCII</td>
</tr>
</tbody>
</table>

**OPTIONS**

The following options are supported:

- `-c` Displays the output in a canonical format. This format is identical to the OCTET format with a granularity of 1.
-i interface Specifies the interface to retrieve values for DHCP parameters from. If this option is not specified, the primary interface is used.

-n limit Limits the list of values displayed to limit lines.

OPERANDS The following operands are supported:

code Numeric code for the requested DHCP parameter, as defined by the DHCP specification. Vendor options are specified by adding 256 to the actual vendor code.

identifier Mnemonic symbol for the requested DHCP parameter, as listed in dhcp_init(4).

EXIT STATUS The following exit values are returned:

0 Successful operation.

2 The operation was not successful. The DHCP client daemon may not be running, the interface might have failed to configure, or no satisfactory DHCP responses were received.

3 Bad arguments.

4 The operation timed out.

6 Some system error (should never occur).

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsr</td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Evolving</td>
</tr>
</tbody>
</table>

SEE ALSO dhcpagent(1M), ifconfig(1M), init(1M), dhcp_init(4), attributes(5)

diff(1)

NAME    | diff – compare two files
SYNOPSIS| diff [-bitw] [-c | -e | -f | -h | -n | -u] file1 file2
          | diff [-bitw] [-C number] -U number] file1 file2
          | diff [-bitw] [-D string] file1 file2
          |          | directory1 directory2

DESCRIPTION The diff utility will compare the contents of file1 and file2 and write to standard output a list of changes necessary to convert file1 into file2. This list should be minimal. Except in rare circumstances, diff finds a smallest sufficient set of file differences. No output will be produced if the files are identical.

The normal output contains lines of these forms:
n1 a n3,n4
n1,n2 d n3
n1,n2 c n3,n4

where n1 and n2 represent lines file1 and n3 and n4 represent lines in file2 These lines resemble ed(1) commands to convert file1 to file2. By exchanging a for d and reading backward, file2 can be converted to file1. As in ed, identical pairs, where n1=n2 or n3=n4, are abbreviated as a single number.

Following each of these lines come all the lines that are affected in the first file flagged by ‘<’, then all the lines that are affected in the second file flagged by ‘>’.

OPTIONS The following options are supported:
-b      | Ignores trailing blanks (spaces and tabs) and treats other strings of blanks as equivalent.
-i      | Ignores the case of letters. For example, ‘A’ will compare equal to ‘a’.
-t      | Expands TAB characters in output lines. Normal or -c output adds character(s) to the front of each line that may adversely affect the indentation of the original source lines and make the output lines difficult to interpret. This option will preserve the original source’s indentation.
-w      | Ignores all blanks (SPACE and TAB characters) and treats all other strings of blanks as equivalent. For example, ‘if ( a = = b )’ will compare equal to ‘if (a == b)’.

The following options are mutually exclusive:
-c      | Produces a listing of differences with three lines of context. With this option, output format is modified slightly. That is, output begins with identification of the files involved and their creation dates, then each change is separated by a line with a dozen *’s.
The lines removed from file1 are marked with ‘—’. The lines added to file2 are marked ‘+’. Lines that are changed from one file to the other are marked in both files with ‘!’.

```
-C number
```

Produces a listing of differences identical to that produced by -c with number lines of context.

```
-D string
```

Creates a merged version of file1 and file2 with C preprocessor controls included so that a compilation of the result without defining string is equivalent to compiling file1, while defining string will yield file2.

```
-e
```

Produces a script of only a, c, and d commands for the editor ed, which will recreate file2 from file1. In connection with the -e option, the following shell program may help maintain multiple versions of a file. Only an ancestral file ($1) and a chain of version-to-version ed scripts ($2,$3,...) made by diff need be on hand. A “latest version” appears on the standard output.

```
(shift; cat $*; echo '1,$p') | ed - $1
```

```
-f
```

Produces a similar script, not useful with ed, in the opposite order.

```
-h
```

Does a fast, half-hearted job. It works only when changed stretches are short and well separated, but does work on files of unlimited length. Options -c, -C, -D, -e, -f, and -n are unavailable with -h. diff does not descend into directories with this option.

```
-n
```

Produces a script similar to -e, but in the opposite order and with a count of changed lines on each insert or delete command.

```
-u
```

Produces a listing of differences with three lines of context. The output is similar to that of the -c option, except that the context is “unified”. Removed and changed lines in file1 are marked by a ‘-’ while lines added or changed in file2 are marked by a ‘+’. Both versions of changed lines appear in the output, while added, removed, and context lines appear only once. The identification of file1 and file2 is different, with “——” and “+++” being printed where “••••” and “——” would appear with the -c option. Each change is separated by a line of the form

```
⊗ ⊗ -n1,n2 +n3,n4 ⊗
```

```
-U number
```

Produces a listing of differences identical to that produced by -u with number lines of context.

The following options are used for comparing directories:

```
-l
```

Produces output in long format. Before the diff, each text file is piped through pr(1) to paginate it. Other differences are remembered and summarized after all text file differences are reported.
Applies `diff` recursively to common subdirectories encountered.

- `s` Reports files that are the identical. These identical files would not otherwise be mentioned.

- `S name` Starts a directory `diff` in the middle, beginning with the file `name`.

**OPERANDS**

The following operands are supported:

- `file1` A path name of a file or directory to be compared. If either `file1` or `file2` is `−`, the standard input will be used in its place.

- `file2` A path name of a file or directory to be compared.

- `directory1` A path name of a directory to be compared.

- `directory2` A path name of a directory to be compared.

If only one of `file1` and `file2` is a directory, `diff` will be applied to the non-directory file and the file contained in the directory file with a filename that is the same as the last component of the non-directory file.

**USAGE**

See `largefile(5)` for the description of the behavior of `diff` when encountering files greater than or equal to 2 Gbyte (\(2^{31}\) bytes).

**EXAMPLES**

**EXAMPLE 1** Typical output of the `diff` command

In the following command, `dir1` is a directory containing a directory named `x`, `dir2` is a directory containing a directory named `x`, `dir1/x` and `dir2/x` both contain files named `date.out`, and `dir2/x` contains a file named `y`:

```
example% diff -r dir1 dir2
Common subdirectories: dir1/x and dir2/x

Only in dir2/x: y

diff -r dir1/x/date.out dir2/x/date.out
1c1
< Mon Jul 2 13:12:16 PDT 1990
---
> Tue Jun 19 21:41:39 PDT 1990
```

**ENVIRONMENT VARIABLES**

See `environ(5)` for descriptions of the following environment variables that affect the execution of `diff`: `LC_CTYPE`, `LC_MESSAGES`, `LC_TIME`, and `NLSPATH`.

- `TZ` Determines the locale for affecting the timezone used for calculating file timestamps written with the `−C` and `−c` options.

**EXIT STATUS**

The following exit values are returned:

- `0` No differences were found.
Differences were found.

An error occurred.

FILES
/tmp/d????? temporary file used for comparison
/usr/lib/diffh executable file for -h option

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

SEE ALSO bdiff(1), cmp(1), comm(1), dircmp(1), ed(1), pr(1), sdiff(1), attributes(5), environ(5), largefile(5)

NOTES Editing scripts produced under the -e or -f options are naive about creating lines consisting of a single period (.).

Missing NEWLINE at end of file indicates that the last line of the file in question did not have a NEWLINE. If the lines are different, they will be flagged and output, although the output will seem to indicate they are the same.
diff3(1)

NAME    diff3 – 3-way differential file comparison

SYNOPSIS    diff3 [-eEX3] filename1 filename2 filename3

DESCRIPTION    diff3 compares three versions of a file, and publishes disagreeing ranges of text flagged with these codes:

-====-            all three files differ
-====1             filename1 is different
-====2             filename2 is different
-====3             filename3 is different

The type of change suffered in converting a given range of a given file to some other is indicated in one of these ways:

\[f : n1 a\]  Text is to be appended after line number \(n1\) in file \(f\), where \(f = 1\), \(2\), or \(3\).

\[f : n1 , n2 c\]  Text is to be changed in the range line \(n1\) to line \(n2\). If \(n1 = n2\), the range may be abbreviated to \(n1\).

The original contents of the range follows immediately after a \(c\) indication. When the contents of two files are identical, the contents of the lower-numbered file is suppressed.

The following command will apply the resulting script to \(filename1\).

\[(\text{cat script; echo } '1,\$p') | \text{ed} - filename1\]

OPTIONS

- \(e\)  Produce a script for the \text{ed}(1) editor that will incorporate into \(filename1\) all changes between \(filename2\) and \(filename3\) (that is, the changes that normally would be flagged \(-====\) and \(-====3\)).

- \(x\)  Produce a script to incorporate only changes flagged \(-====\).

- \(3\)  Produce a script to incorporate only changes flagged \(-====3\).

- \(E\)  Produce a script that will incorporate all changes between \(filename2\) and \(filename3\), but treat overlapping changes (that is, changes that would be flagged with \(-====\) in the normal listing) differently. The overlapping lines from both files will be inserted by the edit script, bracketed by \(<<<<<<\) and \(>>>>>>\) lines.

- \(X\)  Produce a script that will incorporate only changes flagged \(-====\), but treat these changes in the manner of the \(-E\) option.

USAGE    See largefile(5) for the description of the behavior of diff3 when encountering files greater than or equal to 2 Gbyte (\(2^{31}\) bytes).
FILES
/tmp/d3*
/usr/lib/diff3prog

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

SEE ALSO
diff(1), attributes(5), largefile(5)

NOTES
Text lines that consist of a single ‘.’ will defeat -e.

Files longer than 64 Kbytes will not work.
diffmk(1)

NAME
diffmk – mark differences between versions of a troff input file

SYNOPSIS
diffmk oldfile newfile markedfile

DESCRIPTION
diffmk compares two versions of a file and creates a third version that includes
“change mark” (.mc) commands for nroff(1) and troff(1). oldfile and newfile are the
old and new versions of the file. diffmk generates markedfile, which, contains the text
from newfile with troff(1) “change mark” requests (.mc) inserted where newfile
differs from oldfile. When markedfile is formatted, changed or inserted text is shown by
| at the right margin of each line. The position of deleted text is shown by a single *.

USAGE
See largefile(5) for the description of the behavior of diffmk when encountering
files greater than or equal to 2 Gbyte (2^{31} bytes).

EXAMPLES
example

diffmk can also be used in conjunction with the proper troff requests to produce
program listings with marked changes. In the following command line:

diffmk old.c new.c marked.c ; nroff reqs marked.c | pr

the file reqs contains the following troff requests:

.pl 1
.ll 77
.nf
.eo
.nh

which eliminate page breaks, adjust the line length, set no-fill mode, ignore escape
characters, and turn off hyphenation, respectively.

If the characters | and * are inappropriate, you might run markedfile through sed(1) to
globally change them.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWdoc</td>
</tr>
</tbody>
</table>

SEE ALSO
diff(1), nroff(1), sed(1), troff(1), attributes(5), largefile(5)

BUGS
Aesthetic considerations may dictate manual adjustment of some output. File
differences involving only formatting requests may produce undesirable output, that
is, replacing .sp by .sp 2 will produce a “change mark” on the preceding or
following line of output.
dircmp – directory comparison

dircmp [-ds] [-w n] dir1 dir2

The dircmp command examines dir1 and dir2 and generates various tabulated information about the contents of the directories. Listings of files that are unique to each directory are generated for all the options. If no option is entered, a list is output indicating whether the file names common to both directories have the same contents.

The following options are supported:
-\(d\) \hspace{1em} Compares the contents of files with the same name in both directories and output a list telling what must be changed in the two files to bring them into agreement. The list format is described in diff(1).
-\(s\) \hspace{1em} Suppresses messages about identical files.
-\(w n\) \hspace{1em} Changes the width of the output line to \(n\) characters. The default width is 72.

The following operands are supported:
dir1
\hspace{1em} A path name of a directory to be compared.
dir2

See largefile(5) for the description of the behavior of dircmp when encountering files greater than or equal to 2 Gbyte \(\left(2^{31}\right)\) bytes.

See environ(5) for descriptions of the following environment variables that affect the execution of dircmp: LC_COLLATE, LC_CTYPE, LC_MESSAGES, and NLSPATH.

The following exit values are returned:
0 \hspace{1em} Successful completion.
>0 \hspace{1em} An error occurred. (Differences in directory contents are not considered errors.)

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
</tbody>
</table>

See also cmp(1), diff(1), attributes(5), environ(5), largefile(5)
dis – object code disassembler

### SYNOPSIS

```bash
[-l string] [-t sec] file...
```

### DESCRIPTION

The `dis` command produces an assembly language listing of `file`, which may be an object file or an archive of object files. The listing includes assembly statements and an octal or hexadecimal representation of the binary that produced those statements. However, the IA64 listing is limited to assembly statements only.

### OPTIONS

The following options are interpreted by the disassembler and may be specified in any order.

- `-C` Displays demangled C++ symbol names in the disassembly.
- `-d sec` Disassembles the named section as data, printing the offset of the data from the beginning of the section.
- `-D sec` Disassembles the named section as data, printing the actual address of the data.
- `-F function` Disassembles only the named function in each object file specified on the command line. The `-F` option may be specified multiple times on the command line.
- `-l string` Disassembles the archive file specified by `string`. For example, one would issue the command `dis -l x -l z` to disassemble `libx.a` and `libz.a`, which are assumed to be in `LIBDIR`.
- `-L` Invokes a lookup of C-language source labels in the symbol table for subsequent writing to standard output.
- `-o` Prints numbers in octal. The default is hexadecimal.
- `-t sec` Disassembles the named section as text.
- `-V` Prints, on standard error, the version number of the disassembler being executed.

If the `-d`, `-D`, or `-t` options are specified, only those named sections from each user-supplied file will be disassembled. Otherwise, all sections containing text will be disassembled.

On output, a number enclosed in brackets at the beginning of a line, such as `[5]`, indicates that the break-pointable line number starts with the following instruction. These line numbers will be printed only if the file was compiled with additional debugging information, for example, the `-g` option of `cc(1B)`. An expression such as `<40>` in the operand field or in the symbolic disassembly, following a relative displacement for control transfer instructions, is the computed address within the section to which control will be transferred. A function name will appear in the first column, followed by `()` if the object file contains a symbol table.

### OPERANDS

The following operand is supported:

<table>
<thead>
<tr>
<th>OPERANDS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dis</code></td>
<td>Object code disassembler</td>
</tr>
</tbody>
</table>
**file**

A path name of an object file or an archive (see ar(1)) of object files.

**ENVIRONMENT VARIABLES**

See environ(5) for descriptions of the following environment variables that affect the execution of dis: LC_CTYPE, LC_MESSAGES, and NLSPATH.

**LIBDIR**

If this environment variable contains a value, use this as the path to search for the library. If the variable contains a null value, or is not set, it defaults to searching for the library under /usr/lib.

**EXIT STATUS**

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

**FILES**

/usr/lib default LIBDIR

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWbtool</td>
</tr>
</tbody>
</table>

**SEE ALSO**

ar(1), as(1), cc(1B), ld(1), a.out(4), attributes(5), environ(5)

**DIAGNOSTICS**

The self-explanatory diagnostics indicate errors in the command line or problems encountered with the specified files.
dispgid(1)

NAME  dispgid – displays a list of all valid group names

SYNOPSIS  dispgid

DESCRIPTION  dispgid displays a list of all group names on the system (one group per line).

EXIT STATUS  The following exit values are returned:

0  Successful execution.
1  Cannot read the group file.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO  attributes(5)
dispuid(1)

NAME
dispuid – displays a list of all valid user names

SYNOPSIS
dispuid

DESCRIPTION
dispuid displays a list of all user names on the system (one line per name).

EXIT STATUS
The following exit values are returned:

0  Successful execution.
1  Cannot read the password file.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO
attributes(5)
NAME
dos2unix - convert text file from DOS format to ISO format

SYNOPSIS

DESCRIPTION
The dos2unix utility converts characters in the DOS extended character set to the corresponding ISO standard characters.

This command can be invoked from either DOS or SunOS. However, the filenames must conform to the conventions of the environment in which the command is invoked.

If the original file and the converted file are the same, dos2unix will rewrite the original file after converting it.

OPTIONS
The following options are supported:

-ascii  Removes extra carriage returns and converts end of file characters in DOS format text files to conform to SunOS requirements.

-iso    This is the default. It converts characters in the DOS extended character set to the corresponding ISO standard characters.

-7      Converts 8 bit DOS graphics characters to 7 bit space characters so that SunOS can read the file.

On non-i386 systems, dos2unix will attempt to obtain the keyboard type to determine which code page to use. Otherwise, the default is US. The user may override the code page with one of the following options:

-437    Use US code page
-850    Use multilingual code page
-860    Use Portuguese code page
-863    Use French Canadian code page
-865    Use Danish code page

OPERANDS
The following operands are required:

originalfile  The original file in DOS format that is being converted to ISO format.

convertedfile  The new file in ISO format that has been converted from the original DOS file format.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
</tbody>
</table>
File filename not found, or no read permission
The input file you specified does not exist, or you do not have read permission.
Check with the SunOS command, `ls -l` (see `ls(1)`).

Bad output filename filename, or no write permission
The output file you specified is either invalid, or you do not have write permission for that file or the directory that contains it. Check also that the drive or diskette is not write-protected.

Error while writing to temporary file
An error occurred while converting your file, possibly because there is not enough space on the current drive. Check the amount of space on the current drive using the DIR command. Also be certain that the default diskette or drive is write-enabled (not write-protected). Notice that when this error occurs, the original file remains intact.

Translated temporary file name = filename.
Could not rename temporary file to filename.
The program could not perform the final step in converting your file. Your converted file is stored under the name indicated on the second line of this message.

SEE ALSO
unix2dos(1), ls(1), attributes(5)

DIAGNOSTICS
download(1)

NAME
download – host resident PostScript font downloader

SYNOPSIS
download [-f] [-p printer] [-m name] [-H directory] [file...]
/usr/lib/lp/postscript/download

DESCRIPTION
download prepends host resident fonts to files and writes the results on the standard output. If no files are specified, or if – is one of the input files, the standard input is read. download assumes the input files make up a single PostScript job and that requested fonts can be included at the start of each input file.

Requested fonts are named in a comment (marked with %DocumentFonts:) in the input files. Available fonts are the ones listed in the map table selected using the -m option.

The map table consists of fontname–file pairs. The fontname is the full name of the PostScript font, exactly as it would appear in a %DocumentFonts: comment. The file is the pathname of the host resident font. A file that begins with a / is used as is. Otherwise the pathname is relative to the host font directory. Comments are introduced by % (as in PostScript) and extend to the end of the line.

The only candidates for downloading are fonts listed in the map table that point to readable files. A font is downloaded once, at most. Requests for unlisted fonts or inaccessible files are ignored. All requests are ignored if the map table cannot be read.

OPTIONS
-f Force a complete scan of each input file. In the absence of an explicit comment pointing download to the end of the file, the default scan stops immediately after the PostScript header comments.
-p printer Check the list of printer-resident fonts in /etc/lp/printers/printer/residentfonts before downloading.
-m name Use name as the font map table. A name that begins with / is the full pathname of the map table and is used as is. Otherwise name is appended to the pathname of the host font directory.
-H directory Use dir as the host font directory. The default is /usr/lib/lp/postscript.

EXAMPLES

EXAMPLE 1 Examples of the download command.

The following map table could be used to control the downloading of the Bookman font family:

```
% The first string is the full PostScript font name. The second string
% is the file name - relative to the host font directory unless it begins
% with a /.
```

```
%Bookman: /usr/lib/lp/postscript/download
```
EXAMPLE 1 Examples of the download command. (Continued)

<table>
<thead>
<tr>
<th>Font Style</th>
<th>Filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bookman-Light</td>
<td>bookman/light</td>
</tr>
<tr>
<td>Bookman-LightItalic</td>
<td>bookman/lightitalic</td>
</tr>
<tr>
<td>Bookman-Demi</td>
<td>bookman/demi</td>
</tr>
<tr>
<td>Bookman-DemiItalic</td>
<td>bookman/demiitalic</td>
</tr>
</tbody>
</table>

Using the file myprinter/map (in the default host font directory) as the map table, you could download fonts by issuing the following command:

```bash
eexample% download -m myprinter/map file
```

EXIT STATUS The following exit values are returned:

- **0** Successful completion.
- **non-zero** An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWpsf</td>
</tr>
</tbody>
</table>

SEE ALSO dpost(1), postdaisy(1), postmd(1), postio(1), postmd(1), postprint(1), posttek(1), attributes(5)

NOTES The download program should be part of a more general program.

- download does not look for %%PageFonts: comments and there is no way to force multiple downloads of a particular font.

- Using full pathnames in either map tables or the names of map tables is not recommended.
dpost(1)

NAME
dpost – troff postprocessor for PostScript printers

SYNOPSIS
dpost [-c num] [-e num] [-m num] [-n num] [-o list] [-w num]
/usr/lib/lp/postscript/dpost

DESCRIPTION
dpost translates files created by troff(1) into PostScript and writes the results on the
standard output. If no files are specified, or if – is one of the input files, the standard
input is read.

The files should be prepared by troff. The default font files in
/usr/lib/font/devpost produce the best and most efficient output. They assume
a resolution of 720 dpi, and can be used to format files by adding the -Tpost option
to the troff call. Older versions of the eqn and pic preprocessors need to know the
resolution that troff will be using to format the files. If those are the versions
installed on your system, use the -r720 option with eqn and -T720 with pic.

dpost makes no assumptions about resolutions. The first xres command sets the
resolution used to translate the input files, the DESC.out file, usually
/usr/lib/font/devpost/DESC.out, defines the resolution used in the binary font
files, and the PostScript prologue is responsible for setting up an appropriate user
coordinate system.

OPTIONS
-c num        Print num copies of each page. By default only one copy is printed.
-e num        Sets the text encoding level to num. The recognized choices are 0, 1,
              and 2. The size of the output file and print time should decrease as
              num increases. Level 2 encoding will typically be about 20 percent
              faster than level 0, which is the default and produces output
              essentially identical to previous versions of dpost.
-m num        Magnify each logical page by the factor num. Pages are scaled
              uniformly about the origin, which is located near the upper left
corner of each page. The default magnification is 1.0.
-n num        Print num logical pages on each piece of paper, where num can be
              any positive integer. By default, num is set to 1.
-o list       Print those pages for which numbers are given in the
              comma-separated list. The list contains single numbers N and
              ranges N1–N2. A missing N1 means the lowest numbered page, a
              missing N2 means the highest. The page range is an expression of
              logical pages rather than physical sheets of paper. For example, if
              you are printing two logical pages to a sheet, and you specified a
              range of 4, then two sheets of paper would print, containing four
              page layouts. If you specified a page range of 3–4, when
              requesting two logical pages to a sheet; then only page 3 and page
              4 layouts would print, and they would appear on one physical
              sheet of paper.
-p mode  Print files in either portrait or landscape mode. Only the first character of mode is significant. The default mode is portrait.

-w num  Set the line width used to implement troff graphics commands to num points, where a point is approximately 1/72 of an inch. By default, num is set to 0.3 points.

-x num  Translate the origin num inches along the positive x axis. The default coordinate system has the origin fixed near the upper left corner of the page, with positive x to the right and positive y down the page. Positive num moves everything right. The default offset is 0 inches.

-y num  Translate the origin num inches along the positive y axis. Positive num moves text up the page. The default offset is 0.

-F dir  Use dir as the font directory. The default dir is /usr/lib/font, and dpost reads binary font files from directory /usr/lib/font/devpost.

-H dir  Use dir as the host resident font directory. Files in this directory should be complete PostScript font descriptions, and must be assigned a name that corresponds to the appropriate two-character troff font name. Each font file is copied to the output file only when needed and at most once during each job. There is no default directory.

-L file  Use file as the PostScript prologue which, by default, is /usr/lib/lp/postscript/dpost.ps.

-O      Disables PostScript picture inclusion. A recommended option when dpost is run by a spooler in a networked environment.

-T name  Use font files for device name as the best description of available PostScript fonts. By default, name is set to post and dpost reads binary files from /usr/lib/font/devpost.

EXAMPLES  EXAMPLE 1 Examples of the dpost command.

If the old versions of eqn and pic are installed on your system, you can obtain the best possible looking output by issuing a command line such as the following:

```
example% pic -T720 file | tbl | eqn -r720 | troff -mm -Tpost | dpost
```

Otherwise,

```
example% pic file | tbl | eqn | troff -mm -Tpost | dpost
```

should give the best results.

EXIT STATUS  The following exit values are returned:

0       Successful completion.
dpost(1)

non-zero    An error occurred.

FILES
/usr/lib/font/devpost/*.out
/usr/lib/font/devpost/charlib/*
/usr/lib/lp/postscript/color.ps
/usr/lib/lp/postscript/draw.ps
/usr/lib/lp/postscript/forms.ps
/usr/lib/lp/postscript/ps.requests
/usr/lib/macros/pictures
/usr/lib/macros/color

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWpsf</td>
</tr>
</tbody>
</table>

SEE ALSO
download(1), postdaisy(1), postmd(1), postio(1), postmd(1), postprint(1), postreverse(1), posttek(1), troff(1), attributes(5)

NOTES
Output files often do not conform to Adobe’s file structuring conventions. Piping the output of dpost through postreverse(1) should produce a minimally conforming PostScript file.

Although dpost can handle files formatted for any device, emulation is expensive and can easily double the print time and the size of the output file. No attempt has been made to implement the character sets or fonts available on all devices supported by troff. Missing characters will be replaced by white space, and unrecognized fonts will usually default to one of the Times fonts (that is, R, I, B, or BI).

An x res command must precede the first x init command, and all the input files should have been prepared for the same output device.

Use of the -T option is not encouraged. Its only purpose is to enable the use of other PostScript font and device description files, that perhaps use different resolutions, character sets, or fonts.

Although level 0 encoding is the only scheme that has been thoroughly tested, level 2 is fast and may be worth a try.
NAME  du – summarize disk usage
SYNOPSIS  
/usr/bin/du  [-adLr] [-k | -h] [-o | -s] [file ...]
/usr/xpg4/bin/du  [-a | -s] [-k | -h] [-rx] [file ...]

DESCRIPTION  The du utility writes to standard output the size of the file space allocated to, and the size of the file space allocated to each subdirectory of, the file hierarchy rooted in each of the specified files. The size of the file space allocated to a file of type directory is defined as the sum total of space allocated to all files in the file hierarchy rooted in the directory plus the space allocated to the directory itself. This sum will include the space allocated to any extended attributes encountered.

Files with multiple links will be counted and written for only one entry. The directory entry that is selected in the report is unspecified. By default, file sizes are written in 512-byte units, rounded up to the next 512-byte unit.

/usr/xpg4/bin/du  When du cannot obtain file attributes or read directories (see stat(2)), it will report an error condition and the final exit status will be affected.

OPTIONS  The following options are supported for /usr/bin/du and /usr/xpg4/bin/du:

- a  In addition to the default output, report the size of each file not of type directory in the file hierarchy rooted in the specified file. Regardless of the presence of the - a option, non-directories given as file operands will always be listed.

- h  All sizes are scaled to a human readable format, for example, 14K, 234M, 2.7G, or 3.0T. Scaling is done by repetitively dividing by 1024.

- k  Write the files sizes in units of 1024 bytes, rather than the default 512-byte units.

- s  Instead of the default output, report only the total sum for each of the specified files.

/usr/bin/du  The following options are supported for /usr/bin/du only:

- d  Do not cross filesystem boundaries. For example, du - d / reports usage only on the root partition.

- L  Process symbolic links by using the file or directory which the symbolic link references, rather than the link itself.

- o  Do not add child directories’ usage to a parent’s total. Without this option, the usage listed for a particular directory is the space taken by the files in that directory, as well as the files in all directories beneath it. This option does nothing if - s is used.

- r  Generate messages about directories that cannot be read, files that cannot be opened, and so forth, rather than being silent (the default).

/usr/xpg4/bin/du  The following options are supported for /usr/xpg4/bin/du only:
By default, generate messages about directories that cannot be read, files that cannot be opened, and so forth.

When evaluating file sizes, evaluate only those files that have the same device as the file specified by the file operand.

The following operand is supported:

file  The path name of a file whose size is to be written. If no file is specified, the current directory is used.

The output from du consists of the amount of the space allocated to a file and the name of the file.

See largefile(5) for the description of the behavior of du when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

See environ(5) for descriptions of the following environment variables that affect the execution of du: LC_CTYPE, LC_MESSAGES, and NLSPATH.

The following exit values are returned:

0  Successful completion.

>0  An error occurred.

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

<table>
<thead>
<tr>
<th>/usr/bin/du</th>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
<td></td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
<td></td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Stable</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>/usr/xpg4/bin/du</th>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWxcu4</td>
<td></td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
<td></td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Standard</td>
<td></td>
</tr>
</tbody>
</table>

SEE ALSO  ls(1), stat(2), attributes(5), environ(5), fsattr(5), largefile(5), XPG4(5)

System Administration Guide: Basic Administration

NOTES  A file with two or more links is counted only once. If, however, there are links between files in different directories where the directories are on separate branches of the file system hierarchy, du will count the excess files more than once.
Files containing holes will result in an incorrect block count.
du(1B)

NAME
du – display the number of disk blocks used per directory or file

SYNOPSIS
/usr/ucb/du [-adkLr] [-o | -s] [filename]

DESCRIPTION
The du utility gives the number of kilobytes contained in all files and, recursively, directories within each specified directory or file filename. If filename is missing, ‘.’ (the current directory) is used.

A file that has multiple links to it is only counted once.

OPTIONS
The following options are supported:

-a Generates an entry for each file.
-d Does not cross file system boundaries. For example, du -d / reports usage only on the root partition.
-k Writes the files sizes in units of 1024 bytes, rather than the default 512-byte units.
-L Processes symbolic links by using the file or directory that the symbolic link references, rather than the link itself.
-o Does not add child directories’ usage to a parent’s total. Without this option, the usage listed for a particular directory is the space taken by the files in that directory, as well as the files in all directories beneath it. This option does nothing if the -s option is used.
-r Generates messages about directories that cannot be read, files that cannot be opened, and so forth, rather than being silent (the default).
-s Only displays the grand total for each of the specified filenames.

Entries are generated only for each directory in the absence of options.

EXAMPLES
EXAMPLE 1 Showing usage of all subdirectories in a directory

This example uses du in a directory. The pwd(1) command was used to identify the directory, then du was used to show the usage of all the subdirectories in that directory. The grand total for the directory is the last entry in the display:

```bash
eexample$ pwd
/usr/ralph/misc
example$ du
5 ./jokes
33 ./squash
44 ./tech.papers/lpr.document
217 ./tech.papers/new.manager
401 ./tech.papers
144 ./memos
80 ./letters
388 ./window
93 ./messages
15 ./useful.news
1211 .
```
If any of the LC_* variables, that is, LC_CTYPE, LC_MESSAGES, LC_TIME, LC_COLLATE, LC_NUMERIC, and LC_MONETARY (see environ(5)), are not set in the environment, the operational behavior of du for each corresponding locale category is determined by the value of the LANG environment variable. If LC_ALL is set, its contents are used to override both the LANG and the other LC_* variables. If none of the above variables is set in the environment, the "C" (U.S. style) locale determines how du behaves.

**LC_CTYPE**
Determines how du handles characters. When LC_CTYPE is set to a valid value, du can display and handle text and filenames containing valid characters for that locale. du can display and handle Extended Unix Code (EUC) characters where any individual character can be 1, 2, or 3 bytes wide. du can also handle EUC characters of 1, 2, or more column widths. In the "C" locale, only characters from ISO 8859-1 are valid.

**LC_MESSAGES**
Determines how diagnostic and informative messages are presented. This includes the language and style of the messages, and the correct form of affirmative and negative responses. In the "C" locale, the messages are presented in the default form found in the program itself (in most cases, U.S. English).

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

**SEE ALSO**
pwd(1), df(1M), du(1), quot(1M), attributes(5), environ(5)

**NOTES**
Filename arguments that are not directory names are ignored, unless you use -a.
If there are too many distinct linked files, du will count the excess files more than once.
dump(1)

NAME
dump – dump selected parts of an object file

SYNOPSIS
dump [-aCcfghLorstV] [-T index [index]] filename...
dump [-afhorstL [v]] filename...
dump [-hsr [-d number [number]]] filename...
dump [-hsrt [-n name]] filename...

DESCRIPTION
The dump utility dumps selected parts of each of its object file arguments.

The dump utility is best suited for use in shell scripts, whereas the elfdump(1)
command is recommended for more human-readable output.

OPTIONS
This utility will accept both object files and archives of object files. It processes each
file argument according to one or more of the following options:

- a
  Dumps the archive header of each member of an archive.
- c
  Dumps the string table(s).
- C
  Dumps decoded C++ symbol table names.
- f
  Dumps each file header.
- g
  Dumps the global symbols in the symbol table of an archive.
- h
  Dumps the section headers.
- L
  Dumps dynamic linking information and static shared library information, if available.
- o
  Dumps each program execution header.
- r
  Dumps relocation information.
- s
  Dumps section contents in hexadecimal.
- t
  Dumps symbol table entries.
- T index
- T index1,index2
  Dumps only the indexed symbol table entry defined by index or a range of entries defined by index1, index2.
- V
  Prints version information.

The following modifiers are used in conjunction with the options listed above to
modify their capabilities.

- d number
- d number1,number2
  Dumps the section number indicated by number or the range of sections starting at number1 and ending at
  number2. This modifier can be used with -h, -s, and -r. When -d is used with -h or -s, the argument is
dump(1)

treated as the number of a section or range of sections. When -d is used with -r, the argument is treated as the number of the section or range of sections to which the relocation applies. For example, to print out all relocation entries associated with the .text section, specify the number of the section as the argument to -d. If .text is section number 2 in the file, dump -r -d 2 will print all associated entries. To print out a specific relocation section, use dump -s -n name for raw data output, or dump -sv -n name for interpreted output.

-n name

Dumps information pertaining only to the named entity. This modifier can be used with -h, -s, -r, and -t. When -n is used with -h or -s, the argument will be treated as the name of a section. When -n is used with -t or -r, the argument will be treated as the name of a symbol. For example, dump -t -n .text will dump the symbol table entry associated with the symbol whose name is .text, where dump -h -n .text will dump the section header information for the .text section.

-p

Suppresses printing of the headings.

-v

Dumps information in symbolic representation rather than numeric. This modifier can be used with

- a  (date, user id, group id)
- f  (class, data, type, machine, version, flags)
- h  (type, flags)
- L  (value)
- o  (type, flags)
- r  (name, type)
- s  (interpret section contents wherever possible)
- t  (type, bind)

When -v is used with -s, all sections that can be interpreted, such as the string table or symbol table, will be interpreted. For example, dump -sv -n .syntab filename... will produce the same formatted output as dump -tv filename..., but dump -s -n .syntab filename... will print raw data in hexadecimal. Without additional modifiers, dump -sv filename... will dump
dump(1)

all sections in the files, interpreting all those that it can and dumping the rest (such as .text or .data) as raw data.

The dump utility attempts to format the information it dumps in a meaningful way, printing certain information in character, hexadecimal, octal, or decimal representation as appropriate.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWbtool</td>
</tr>
</tbody>
</table>

SEE ALSO elfdump(1), nm(1), ar(3HEAD), a.out(4), attributes(5)
dumpcs(1)

NAME
dumpcs – show codeset table for the current locale

SYNOPSIS
dumpcs [-012vw]

DESCRIPTION
dumpcs shows a list of printable characters for the user’s current locale, along with their hexadecimal code values. The display device is assumed to be capable of displaying characters for a given locale. With no option, dumpcs displays the entire list of printable characters for the current locale.

With one or more numeric options specified, it shows EUC codeset(s) for the current locale according to the numbers specified, and in order of codeset number. Each non-printable character is represented by an asterisk “*” and enough ASCII space character(s) to fill that codeset’s column width.

OPTIONS
-0 Show ASCII (or EUC primary) codeset.
-1 Show EUC codeset 1, if used for the current locale.
-2 Show EUC codeset 2, if used for the current locale.
-3 Show EUC codeset 3, if used for the current locale.
-v “Verbose”. Normally, ranges of non-printable characters are collapsed into a single line. This option produces one line for each non-printable character.
-w Replace code values with corresponding wide character values (process codes).

ENVIRONMENT VARIABLES
The environment variables LC_CTYPE and LANG control the character classification throughout dumpcs. On entry to dumpcs, these environment variables are checked in that order. This implies that a new setting for LANG does not override the setting of LC_CTYPE. When none of the values is valid, the character classification defaults to the POSIX.1 “C” locale.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO
localedef(1), attributes(5)

NOTES
dumpcs can only handle EUC locales.
echo(1)

NAME    echo – echo arguments

SYNOPSIS /usr/bin/echo [string...]

DESCRIPTION The echo utility writes its arguments, separated by BLANKs and terminated by a
NEWLINE, to the standard output. If there are no arguments, only the NEWLINE
character will be written.

echo is useful for producing diagnostics in command files, for sending known data
into a pipe, and for displaying the contents of environment variables.

The C shell, the Korn shell, and the Bourne shell all have echo built-in commands,
which, by default, will be invoked if the user calls echo without a full pathname. See
shell_builtins(1). sh's echo, ksh's echo, and /usr/bin/echo understand the
back-slashed escape characters, except that sh's echo does not understand \a as the
alert character. In addition, ksh's echo, does not have an -n option. sh's echo and
/usr/bin/echo only have an -n option if the SYSV3 environment variable is set (see
ENVIRONMENT VARIABLES below). If it is, none of the backslashed characters
mentioned above are available. csh's echo and /usr/ucb/echo, on the other hand,
have an -n option, but do not understand the back-slashed escape characters.

OPERANDS The following operand is supported:

string A string to be written to standard output. If any operand is "-n", it will be
treated as a string, not an option. The following character sequences will be
recognized within any of the arguments:

\a Alert character.
\b Backspace.
\c Print line without new-line. All characters following the \c in
the argument are ignored.
\f Form-feed.
\n New-line.
\r Carriage return.
\t Tab.
\v Vertical tab.
\ Backslash.
\on Where \n is the 8-bit character whose ASCII code is the 1-, 2- or
3-digit octal number representing that character.

USAGE Portable applications should not use -n (as the first argument) or escape sequences.

The printf(1) utility can be used portably to emulate any of the traditional behaviors
of the echo utility as follows:
The Solaris 2.6 operating environment or compatible version’s /usr/bin/echo is equivalent to:

```
printf "%b\n" "$*"
```

The /usr/ucb/echo is equivalent to:

```
if [ "X$1" = "X-n" ]
then
    shift
    printf "%s" "$*
else
    printf "%s\n" "$*
fi
```

New applications are encouraged to use printf instead of echo.

**EXAMPLE 1** Finding how far below root your current directory is located

You can use echo to determine how many subdirectories below the root directory (/) is your current directory, as follows:

- Echo your current-working-directory’s full pathname.
- Pipe the output through `tr` to translate the path’s embedded slash-characters into space-characters.
- Pipe that output through `wc -w` for a count of the names in your path.

```
exaempt /usr/bin/echo $PWD | tr '/ ' '' | wc -w
```

See `tr(1)` and `wc(1)` for their functionality.

Below are the different flavors for echoing a string without a NEWLINE:

**EXAMPLE 2** /usr/bin/echo

```
exaempt /usr/bin/echo "$USER’s current directory is $PWD\c"
```

**EXAMPLE 3** sh/ksh shells

```
exaempt echo "$USER’s current directory is $PWD\c"
```

**EXAMPLE 4** csh shell

```
exaempt echo -n "$USER’s current directory is $PWD"
```
**EXAMPLE 5** /usr/ucb/echo

```bash
example$ /usr/ucb/echo -n "$USER's current directory is $PWD"
```

### ENVIRONMENT VARIABLES

**SYSV3**  
This environment variable is used to provide compatibility with INTERACTIVE UNIX System and SCO UNIX installation scripts. It is intended for compatibility only and should not be used in new scripts.

See `environ(5)` for descriptions of the following environment variables that affect the execution of `echo`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

### EXIT STATUS

The following error values are returned:

- **0**  
  Successful completion.

- **>0**  
  An error occurred.

### ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

### SEE ALSO

`echo(1B), printf(1), shell_builtins(1), tr(1), wc(1), ascii(5), attributes(5), environ(5)`

### NOTES

When representing an 8-bit character by using the escape convention `\0n`, the `n` must always be preceded by the digit zero (0).

For example, typing: `echo 'WARNING:\07'` will print the phrase `WARNING:` and sound the “bell” on your terminal. The use of single (or double) quotes (or two backslashes) is required to protect the “\” that precedes the “07”.

Following the `\0`, up to three digits are used in constructing the octal output character. If, following the `\0n`, you want to echo additional digits that are not part of the octal representation, you must use the full 3-digit `n`. For example, if you want to echo “ESC 7” you must use the three digits “033” rather than just the two digits “33” after the `\0`.

<table>
<thead>
<tr>
<th>2 digits</th>
<th>Incorrect: echo’0337 1 od -xc produces: df0a (hex)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>337 (ascii)</td>
</tr>
<tr>
<td>3 digits</td>
<td>Correct: echo ’00337′ 1 od -xc</td>
</tr>
</tbody>
</table>

308  man pages section 1: User Commands • Last Revised 20 Jan 2000
echo(1)

produces:  

\texttt{lb37 0a00} \hspace{1em} \text{(hex)}

\texttt{033 7} \hspace{1em} \text{(ascii)}

For the octal equivalents of each character, see \texttt{ascii(5)}. 

User Commands \hspace{1em} 309
echo – echo arguments to standard output

NAME

SYNOPSIS

DESCRIPTION

echo writes its arguments, separated by BLANKs and terminated by a NEWLINE, to the standard output.

echo is useful for producing diagnostics in command files and for sending known data into a pipe, and for displaying the contents of environment variables.

For example, you can use echo to determine how many subdirectories below the root directory (/) is your current directory, as follows:

- echo your current-working-directory's full pathname
- pipe the output through tr to translate the path’s embedded slash-characters into space-characters
- pipe that output through wc -w for a count of the names in your path.

```
example% /usr/bin/echo "$PWD | tr '/\'' | wc -w"
```

See tr(1) and wc(1) for their functionality.

The shells csh(1), ksh(1), and sh(1), each have an echo built-in command, which, by default, will have precedence, and will be invoked if the user calls echo without a full pathname. /usr/ucb/echo and csh's echo() have an -n option, but do not understand back-slashed escape characters. sh's echo(), ksh's echo(), and /usr/bin/echo, on the other hand, understand the black-slashed escape characters, and ksh's echo() also understands \a as the audible bell character; however, these commands do not have an -n option.

OPTIONS

- n    Do not add the NEWLINE to the output.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

SEE ALSO

csh(1), echo(1), ksh(1), sh(1), tr(1), wc(1), attributes(5)

NOTES

The -n option is a transition aid for BSD applications, and may not be supported in future releases.
echo(1F)

NAME   echo – put string on virtual output

SYNOPSIS echo [string...]

DESCRIPTION The echo function directs each string it is passed to the standard output. If no argument is given, echo looks to the standard input for input. It is often used in conditional execution or for passing a string to another command.

EXAMPLES EXAMPLE 1 A sample of the echo command.

Set the done descriptor to help if a test fails:

done=’if [ -s $F1 ];
    then echo close;
    else echo help;
    fi’

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO echo(1), attributes(5)
The \texttt{ed} utility is the standard text editor. If \texttt{file} is specified, \texttt{ed} simulates an e command (see below) on the named file; that is to say, the file is read into \texttt{ed}'s buffer so that it can be edited.

The \texttt{ed} utility operates on a copy of the file it is editing; changes made to the copy have no effect on the file until a w (write) command is given. The copy of the text being edited resides in a temporary file called the buffer. There is only one buffer.

The \texttt{red} utility is a restricted version of \texttt{ed}. It will only allow editing of files in the current directory. It prohibits executing shell commands via \texttt{!shell command}. Attempts to bypass these restrictions result in an error message (restricted shell).

Both \texttt{ed} and \texttt{red} support the \texttt{fspec(4)} formatting capability. The default terminal mode is either \texttt{stty -tabs} or \texttt{stty tab3}, where tab stops are set at eight columns (see \texttt{stty(1)}). If, however, the first line of \texttt{file} contains a format specification, that specification will override the default mode. For example, if the first line of \texttt{file} contains

\begin{verbatim}
<:t5,10,15 s72:>
\end{verbatim}

the command would be set at 5, 10, and 15, and a maximum line length of 72 would be imposed.

Commands to \texttt{ed} have a simple and regular structure: zero, one, or two \texttt{addresses} followed by a single-character \texttt{command}, possibly followed by parameters to that command. These addresses specify one or more lines in the buffer. Every command that requires addresses has default addresses, so that the addresses can very often be omitted.

In general, only one command may appear on a line. Certain commands allow the input of text. This text is placed in the appropriate place in the buffer. While \texttt{ed} is accepting text, it is said to be in \texttt{input mode}. In this mode, no commands are recognized; all input is merely collected. Leave input mode by typing a period (.) at the beginning of a line, followed immediately by a carriage return.

If \texttt{ed} executes commands with arguments, it uses the default shell \texttt{/usr/bin/sh} (see \texttt{sh(1)}).

If \texttt{ed} executes commands with arguments, it uses \texttt{/usr/xpg4/bin/sh} (see \texttt{ksh(1)}).

The \texttt{ed} utility supports a limited form of \texttt{regular expression} notation. Regular expressions are used in addresses to specify lines and in some commands (for example, s) to specify portions of a line that are to be substituted. To understand
addressing in ed, it is necessary to know that at any time there is a current line. Generally speaking, the current line is the last line affected by a command; the exact effect on the current line is discussed under the description of each command.

Internationalized Basic Regular Expressions are used for all system-supplied locales. See regex(5).

**ed Commands**

Commands may require zero, one, or two addresses. Commands that require no addresses regard the presence of an address as an error. Commands that accept one or two addresses assume default addresses when an insufficient number of addresses is given; if more addresses are given than such a command requires, the last one(s) are used.

Typically, addresses are separated from each other by a comma (,). They may also be separated by a semicolon (;). In the latter case, the first address is calculated, the current line (.) is set to that value, and then the second address is calculated. This feature can be used to determine the starting line for forward and backward searches (see Rules 5 and 6, above). The second address of any two-address sequence must correspond to a line in the buffer that follows the line corresponding to the first address.

In the following list of ed commands, the parentheses shown prior to the command are not part of the address; rather, they show the default address(es) for the command.

Each address component can be preceded by zero or more blank characters. The command letter can be preceded by zero or more blank characters. If a suffix letter (l, n, or p) is given, it must immediately follow the command.

The e, E, f, r, and w commands take an optional file parameter, separated from the command letter by one or more blank characters.

If changes have been made in the buffer since the last w command that wrote the entire buffer, ed will warn the user if an attempt is made to destroy the editor buffer via the e or q commands. The ed utility will write the string:

"?\n"

(followed by an explanatory message if help mode has been enabled via the H command) to standard output and will continue in command mode with the current line number unchanged. If the e or q command is repeated with no intervening command, it will take effect.

If an end-of-file is detected on standard input when a command is expected, the ed utility acts as if a q command had been entered.

It is generally illegal for more than one command to appear on a line. However, any command (except e, f, r, or w) may be suffixed by l, n, or p in which case the current line is either listed, numbered or written, respectively, as discussed below under the l, n, and p commands.
The append command accepts zero or more lines of text and appends it after the addressed line in the buffer. The current line (.) is left at the last inserted line, or, if there were none, at the addressed line. Address 0 is legal for this command: it causes the “appended” text to be placed at the beginning of the buffer. The maximum number of characters that may be entered from a terminal is 256 per line (including the new-line character).

The change command deletes the addressed lines from the buffer, then accepts zero or more lines of text that replaces these lines in the buffer. The current line (.) is left at the last line input, or, if there were none, at the first line that was not deleted; if the lines deleted were originally at the end of the buffer, the current line number will be set to the address of the new last line; if no lines remain in the buffer, the current line number will be set to 0.

Same as the X command, described later, except that ed assumes all text read in for the e and r commands is encrypted unless a null key is typed in.

The delete command deletes the addressed lines from the buffer. The line after the last line deleted becomes the current line; if the lines deleted were originally at the end of the buffer, the new last line becomes the current line. If no lines remain in the buffer, the current line number will be set to 0.

The edit command deletes the entire contents of the buffer and then reads the contents of file into the buffer. The current line (.) is set to the last line of the buffer. If file is not given, the currently remembered file name, if any, is used (see the f command). The number of bytes read will be written to standard output, unless the -s option was specified, in the following format:

```
"%d\ n" <number of bytes read>
```

file is remembered for possible use as a default file name in subsequent e, E, r, and w commands. If file is replaced by !, the rest of the line is taken to be a shell (sh(1)) command whose output is to be read. Such a shell command is not remembered as the current file name. See also DIAGNOSTICS below. All marks will be discarded upon the completion of a successful e command. If the buffer has changed since the last time the entire buffer was written, the user will be warned, as described previously.

The Edit command is like e, except that the editor does not check to see if any changes have been made to the buffer since the last w command.
If `file` is given, the `f` command will change the currently remembered path name to `file`; whether the name is changed or not, it then will write the (possibly new) currently remembered path name to the standard output in the following format:

```
"%s\n"pathname
```

The current line number is unchanged.

```
( 1, $ ) g /RE/command list
```

In the `global` command, the first step is to mark every line that matches the given `RE`. Then, for every such line, the given `command list` is executed with the current line (.) initially set to that line. When the `g` command completes, the current line number will have the value assigned by the last command in the command list. If there were no matching lines, the current line number will not be changed. A single command or the first of a list of commands appears on the same line as the `global` command. All lines of a multi-line list except the last must be ended with a backslash (`\`). `a`, `i`, and `c` commands and associated input are permitted. The terminating input mode may be omitted if it would be the last line of the `command list`. An empty `command list` is equivalent to the `p` command. The `g`, `G`, `v`, `V`, and `!` commands are not permitted in the `command list`. See also the `NOTES` and the last paragraph before `FILES` below. Any character other than space or newline can be used instead of a slash to delimit the `RE`. Within the `RE`, the `RE` delimiter itself can be used as a literal character if it is preceded by a backslash.

```
( 1, $ ) G /RE/
```

In the interactive `global` command, the first step is to mark every line that matches the given `RE`. Then, for every such line, that line is written to standard output, the current line (.) is changed to that line, and any one command (other than one of the `a`, `c`, `i`, `g`, `G`, `v`, and `V` commands) may be input and is executed. After the execution of that command, the next marked line is written, and so on; a new-line acts as a null command; an `&` causes the re-execution of the most recent non-null command executed within the current invocation of `G`. Note: The commands input as part of the execution of the `G` command may address and affect any lines in the buffer. The final value of the current line number will be the value set by the last command successfully executed. (Note that the last command successfully executed will be the `G` command itself if a command fails or the null command is specified.) If there were no matching lines, the current line number will not be changed. The `G` command can be terminated by a `SIGINT` signal. The `G` command can be terminated by an interrupt signal (ASCII DEL or BREAK). Any character other than space or newline can be used instead of a slash to delimit the `RE`. Within the `RE`, the `RE` delimiter itself can be used as a literal character if it is preceded by a backslash.

```
h
```

The `help` command gives a short error message that explains the reason for the most recent `?` diagnostic. The current line number is unchanged.
The `H` command causes `ed` to enter a mode in which error messages are written for all subsequent diagnostics. It will also explain the previous if there was one. The `H` command alternately turns this mode on and off; it is initially off. The current line number is unchanged.

(. )i
<text>
.

The `insert` command accepts zero or more lines of text and inserts it before the addressed line in the buffer. The current line (.) is left at the last inserted line, or, if there were none, at the addressed line. This command differs from the `a` command only in the placement of the input text. Address 0 is not legal for this command. The maximum number of characters that may be entered from a terminal is 256 per line (including the new-line character).

(. , .+1 )j

The `join` command joins contiguous lines by removing the appropriate new-line characters. If exactly one address is given, this command does nothing. If lines are joined, the current line number will be set to the address of the joined line; otherwise, the current line number is unchanged.

(. )k\x

The `mark` command marks the addressed line with name \x, which must be an ASCII lower-case letter (a-z). The address `\x` then addresses this line; the current line (.) is unchanged.

(. , . )l

The `l` command writes to standard output the addressed lines in a visually unambiguous form. The characters (`\`, `a`, `b`, `f`, `r`, `t`, `v`) will be written as the corresponding escape sequence; the `\n` in that table is not applicable. Non-printable characters not in the table will be written as one three-digit octal number (with a preceding backslash character) for each byte in the character (most significant byte first).

Long lines will be folded, with the point of folding indicated by writing backslash/newline character; the length at which folding occurs is unspecified, but should be appropriate for the output device. The end of each line will be marked with a $. An `l` command can be appended to any other command other than `e`, `E`, `f`, `q`, `Q`, `r`, `w`, or `!`. The current line number will be set to the address of the last line written.

(. , . )mv

The `move` command repositions the addressed line(s) after the line addressed by `a`. Address 0 is legal for `a` and causes the addressed line(s) to be moved to the beginning of the file. It is an error if address `a` falls within the range of moved lines; the current line (.) is left at the last line moved.
n
The number command writes the addressed lines, preceding each line by its line number and a tab character; the current line (.) is left at the last line written. The n command may be appended to any command other than e, E, f, q, Q, r, w, or !.

p
The print command writes the addressed lines to standard output; the current line (.) is left at the last line written. The p command may be appended to any command other than e, E, f, q, Q, r, w, or !. For example, dp deletes the current line and writes the new current line.

P
The P command causes ed to prompt with an asterisk (*) (or string, if -p is specified) for all subsequent commands. The P command alternatively turns this mode on and off; it is initially on if the -p option is specified, otherwise off. The current line is unchanged.

q
The quit command causes ed to exit. If the buffer has changed since the last time the entire buffer was written, the user will be warned; see DIAGNOSTICS.

Q
The editor exits without checking if changes have been made in the buffer since the last w command.

$ r file
The read command reads the contents of file into the buffer. If file is not given, the currently remembered file name, if any, is used (see the e and f commands). The currently remembered file name is not changed unless file is the very first file name mentioned since ed was invoked. Address 0 is legal for r and causes the file to be read in at the beginning of the buffer. If the read is successful and the -s option was not specified, the number of characters read is written to standard output in the following format:

%d
<number of bytes read>

The current line (.) is set to the last line read. If file is replaced by !, the rest of the line is taken to be a shell command (see sh(1)) whose output is to be read. For example, $r:ls appends the current directory to the end of the file being edited. Such a shell command is not remembered as the current file name.

s /RE/replacement/
( . , . ) s /RE/replacement /count, count=[1-512]
( . , . ) s /RE/replacement /g
( . , . ) s /RE/replacement /l
( . , . ) s /RE/replacement /n
( . , . ) s /RE/replacement /p
The substitute command searches each addressed line for an occurrence of the specified RE. Zero or more substitution commands can be specified. In each line in which a match is found, all (non-overlapped) matched strings are replaced by the
replacement if the global replacement indicator \textit{g} appears after the command. If the global indicator does not appear, only the first occurrence of the matched string is replaced. If a number \textit{count} appears after the command, only the \textit{count}-th occurrence of the matched string on each addressed line is replaced. It is an error if the substitution fails on all addressed lines. Any character other than space or new-line may be used instead of the slash (/) to delimit the \textit{RE} and the \textit{replacement}; the current line (.) is left at the last line on which a substitution occurred. Within the \textit{RE}, the \textit{RE} delimiter itself can be used as a literal character if it is preceded by a backslash. See also the last paragraph before \textit{FILES} below.

An ampersand (&) appearing in the \textit{replacement} is replaced by the string matching the \textit{RE} on the current line. The special meaning of \& in this context may be suppressed by preceding it by \, As a more general feature, the characters \texttt{\textbackslash n}, where \texttt{n} is a digit, are replaced by the text matched by the \texttt{n}-th regular subexpression of the specified \textit{RE} enclosed between \texttt{\textbackslash (} and \texttt{\textbackslash )}. When nested parenthesized subexpressions are present, \texttt{n} is determined by counting occurrences of \texttt{\textbackslash (} starting from the left. When the character \% is the only character in the \textit{replacement}, the \textit{replacement} used in the most recent substitute command is used as the \textit{replacement} in the current substitute command; if there was no previous substitute command, the use of \% in this manner is an error. The \% loses its special meaning when it is in a replacement string of more than one character or is preceded by a \textbackslash. For each backslash (\) encountered in scanning \textit{replacement} from beginning to end, the following character loses its special meaning (if any). It is unspecified what special meaning is given to any character other than \&, \textbackslash, \%, or digits.

A line may be split by substituting a new-line character into it. The new-line in the \textit{replacement} must be escaped by preceding it by \textbackslash. Such substitution cannot be done as part of a \textit{g} or \textit{v} command list. The current line number will be set to the address of the last line on which a substitution is performed. If no substitution is performed, the current line number is unchanged. If a line is split, a substitution is considered to have been performed on each of the new lines for the purpose of determining the new current line number. A substitution is considered to have been performed even if the replacement string is identical to the string that it replaces.

The substitute command supports the following indicators:

- \textit{count} Substitute for the \texttt{count}-th occurrence only of the \textit{RE} found on each addressed line. \textit{count} must be between \texttt{1-512}.

- \textit{g} Globally substitute for all non-overlapping instances of the \textit{RE} rather than just the first one. If both \textit{g} and \textit{count} are specified, the results are unspecified.

- \textit{l} Write to standard output the final line in which a substitution was made.
The line will be written in the format specified for the \textit{l} command.

- \textit{n} Write to standard output the final line in which a substitution was made.
The line will be written in the format specified for the \textit{n} command.
Write to standard output the final line in which a substitution was made. The line will be written in the format specified for the p command.

This command acts just like the m command, except that a copy of the addressed lines is placed after address a (which may be 0); the current line (.) is left at the last line copied.

The undo command nullifies the effect of the most recent command that modified anything in the buffer, namely the most recent a, c, d, g, i, j, m, r, s, t, u, v, G, or V command. All changes made to the buffer by a g, G, v, or V global command will be undone as a single change; if no changes were made by the global command (such as with g/RE/p), the u command will have no effect. The current line number will be set to the value it had immediately before the command being undone started.

This command is the same as the global command g, except that the lines marked during the first step are those that do not match the RE.

This command is the same as the interactive global command G, except that the lines that are marked during the first step are those that do not match the RE.

The write command writes the addressed lines into file. If file does not exist, it is created with mode 666 (readable and writable by everyone), unless your file creation mask dictates otherwise; see the description of the umask special command on sh(1). The currently remembered file name is not changed unless file is the very first file name mentioned since ed was invoked. If no file name is given, the currently remembered file name, if any, is used (see the e and f commands); the current line (.) is unchanged. If the command is successful, the number of characters written is printed, unless the -s option is specified in the following format:

"%d\n",<number of bytes written>

If file is replaced by !, the rest of the line is taken to be a shell (see sh(1)) command whose standard input is the addressed lines. Such a shell command is not remembered as the current path name. This usage of the write command with ! is to be considered as a "last w command that wrote the entire buffer".

This command is the same as the write command above, except that it appends the addressed lines to the end of file if it exists. If file does not exist, it is created as described above for the w command.

An educated guess is made to determine whether text read for the e and r commands is encrypted. A null key turns off encryption. Subsequent e, r, and w
commands will use this key to encrypt or decrypt the text. An explicitly empty key
turns off encryption. Also, see the -x option of ed.

( $ )=
The line number of the addressed line will be written to standard output in the
following format:

"%d\n"<line number>

The current line number is unchanged by this command.

!shell command
The remainder of the line after the ! is sent to the UNIX system shell (see sh(1)) to
be interpreted as a command. Within the text of that command, the unescaped
character % is replaced with the remembered file name; if a ! appears as the first
character of the shell command, it is replaced with the text of the previous shell
command. Thus, !! will repeat the last shell command. If any replacements of % or
! are performed, the modified line will be written to the standard output before
command is executed. The ! command will write:

"!\n"

to standard output upon completion, unless the -a option is specified. The current
line number is unchanged.

(.+1)<new-line>
An address alone on a line causes the addressed line to be written. A new-line alone
is equivalent to .+1p; it is useful for stepping forward through the buffer. The
current line number will be set to the address of the written line.

If an interrupt signal (ASCII DEL or BREAK) is sent, ed writes a "?\n" and returns to
its command level.

The ed utility will take the standard action for all signals with the following
exceptions:

SIGINT
The ed utility will interrupt its current activity, write the string
"?\n" to standard output, and return to command mode.

SIGHUP
If the buffer is not empty and has changed since the last write, the
ed utility will attempt to write a copy of the buffer in a file. First,
the file named ed.hup in the current directory will be used; if that
fails, the file named ed.hup in the directory named by the HOME
environment variable will be used. In any case, the ed utility will
exit without returning to command mode.

Some size limitations are in effect: 512 characters in a line, 256 characters in a global
command list, and 255 characters in the path name of a file (counting slashes). The
limit on the number of lines depends on the amount of user memory; each line takes 1
word.
When reading a file, ed discards ASCII and NUL characters.

If a file is not terminated by a new-line character, ed adds one and puts out a message explaining what it did.

If the closing delimiter of an RE or of a replacement string (for example, /) would be the last character before a new-line, that delimiter may be omitted, in which case the addressed line is written. The following pairs of commands are equivalent:

\[ s/s1/s2 \quad s/s1/s2/p \]
\[ g/s1 \quad g/s1/p \]
\[ ?s1 \quad ?s1? \]

If an invalid command is entered, ed will write the string:

```
"?\ n"
```

(followed by an explanatory message if help mode has been enabled by the \texttt{H} command) to standard output and will continue in command mode with the current line number unchanged.

**OPTIONS**

- \texttt{-C}  
  Encryption option; the same as the \texttt{-x} option, except that ed simulates a \texttt{c} command. The \texttt{c} command is like the \texttt{x} command, except that all text read in is assumed to have been encrypted.

- \texttt{-p \textit{string}}  
  Allows the user to specify a prompt string. By default, there is no prompt string.

- \texttt{-s | -;}  
  Suppresses the writing of character counts by \texttt{e}, \texttt{r}, and \texttt{w} commands, of diagnostics from \texttt{e} and \texttt{q} commands, and of the \texttt{!} prompt after a \texttt{!shell command}.

- \texttt{-x}  
  Encryption option; when used, ed simulates an \texttt{x} command and prompts the user for a key. The \texttt{x} command makes an educated guess to determine whether text read in is encrypted or not. The temporary buffer file is encrypted also, using a transformed version of the key typed in for the \texttt{-x} option. See \texttt{NOTES}.

**OPERANDS**

The following operand is supported:

\texttt{file}  
If \texttt{file} is specified, ed simulates an \texttt{e} command on the file named by the path name \texttt{file} before accepting commands from the standard input.

**USAGE**

See largefile(5) for the description of the behavior of ed and red when encountering files greater than or equal to 2 Gbyte (\(2^{31}\) bytes).

**ENVIRONMENT VARIABLES**

See environ(5) for descriptions of the following environment variables that affect the execution of ed: \texttt{HOME}, \texttt{LC_CTYPE}, \texttt{LC_COLLATE}, \texttt{LC_MESSAGES}, and \texttt{NLSPATH}.

**EXIT STATUS**

The following exit values are returned:
Successful completion without any file or command errors.

>0 An error occurred.

### FILES

- **$TMPDIR**
  - If this environment variable is not NULL, its value is used in place of `/var/tmp` as the directory name for the temporary work file.

- **/var/tmp**
  - If `/var/tmp` exists, it is used as the directory name for the temporary work file.

- **/tmp**
  - If the environment variable TMPDIR does not exist or is NULL, and if `/var/tmp` does not exist, then `/tmp` is used as the directory name for the temporary work file.

- **ed.hup**
  - Work is saved here if the terminal is hung up.

### ATTRIBUTES

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

#### /usr/bin/red

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

#### /usr/xpg4/bin/ed

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWxcu4</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

### SEE ALSO


### DIAGNOSTICS

- `?` for command errors.
- `?file` for an inaccessible file. (use the help and Help commands for detailed explanations).

If changes have been made in the buffer since the last `w` command that wrote the entire buffer, `ed` warns the user if an attempt is made to destroy `ed`'s buffer via the `e` or `q` commands. It writes `?` and allows one to continue editing. A second `e` or `q` command at this point will take effect. The `-s` command-line option inhibits this feature.

### NOTES

The `-` option, although it continues to be supported, has been replaced in the documentation by the `-s` option that follows the Command Syntax Standard (see intro(1)).

A `!` command cannot be subject to a `g` or a `v` command.

The `!` command and the `!` escape from the `e`, `r`, and `w` commands cannot be used if the editor is invoked from a restricted shell (see `sh(1)`).
The sequence \n in an RE does not match a new-line character.

If the editor input is coming from a command file (for example, ed file < ed_cmd_file), the editor exits at the first failure.
The edit utility is a variant of the text editor ex recommended for new or casual users who wish to use a command-oriented editor. It operates precisely as ex with the following options automatically set:

- `novice`: ON
- `report`: ON
- `showmode`: ON
- `magic`: OFF

The following brief introduction should help you get started with edit. If you are using a CRT terminal you may want to learn about the display editor vi.

To edit the contents of an existing file you begin with the command `edit name` to the shell. edit makes a copy of the file that you can then edit, and tells you how many lines and characters are in the file. To create a new file, you also begin with the command `edit` with a filename: `edit name`; the editor will tell you it is a [New File].

The `edit` command prompt is the colon (:), which you should see after starting the editor. If you are editing an existing file, then you will have some lines in edit’s buffer (its name for the copy of the file you are editing). When you start editing, edit makes the last line of the file the current line. Most commands to edit use the current line if you do not tell them which line to use. Thus if you say `print` (which can be abbreviated `p`) and type carriage return (as you should after all `edit` commands), the current line will be printed. If you `delete` the current line, edit will print the new current line, which is usually the next line in the file. If you `delete` the last line, then the new last line becomes the current one.

If you start with an empty file or wish to add some new lines, then the `append` (a) command can be used. After you execute this command (typing a carriage return after the word `append`), edit will read lines from your terminal until you type a line consisting of just a dot (.). It places these lines after the current line. The last line you type then becomes the current line. The `insert` (i) command is like `append`, but places the lines you type before, rather than after, the current line.

The `edit` utility numbers the lines in the buffer, with the first line having number 1. If you execute the command 1, then `edit` will type the first line of the buffer. If you then execute the command d, `edit` will delete the first line, line 2 will become line 1, and `edit` will print the current line (the new line 1) so you can see where you are. In general, the current line will always be the last line affected by a command.
You can make a change to some text within the current line by using the substitute (s) command: s/old/new/ where old is the string of characters you want to replace and new is the string of characters you want to replace old with.

The filename (f) command will tell you how many lines there are in the buffer you are editing and will say [Modified] if you have changed the buffer. After modifying a file, you can save the contents of the file by executing a write (w) command. You can leave the editor by issuing a quit (q) command. If you run edit on a file, but do not change it, it is not necessary (but does no harm) to write the file back. If you try to quit from edit after modifying the buffer without writing it out, you will receive the message No write since last change (:quit! overrides), and edit will wait for another command. If you do not want to write the buffer out, issue the quit command followed by an exclamation point (!). The buffer is then irretrievably discarded and you return to the shell.

By using the d and a commands and giving line numbers to see lines in the file, you can make any changes you want. You should learn at least a few more things, however, if you will use edit more than a few times.

The change (c) command changes the current line to a sequence of lines you supply (as in append, you type lines up to a line consisting of only a dot (.). You can tell change to change more than one line by giving the line numbers of the lines you want to change, that is, 3,5c. You can print lines this way too: 1,23p prints the first 23 lines of the file.

The undo (u) command reverses the effect of the last command you executed that changed the buffer. Thus if you execute a substitute command that does not do what you want, type u and the old contents of the line will be restored. You can also undo an undo command. edit will give you a warning message when a command affects more than one line of the buffer. Note that commands such as write and quit cannot be undone.

To look at the next line in the buffer, type carriage return. To look at a number of lines, type "^D (while holding down the control key, press d) rather than carriage return. This will show you a half-screen of lines on a CRT or 12 lines on a hardcopy terminal. You can look at nearby text by executing the z command. The current line will appear in the middle of the text displayed, and the last line displayed will become the current line; you can get back to the line where you were before you executed the z command by typing "'. The z command has other options: z− prints a screen of text (or 24 lines) ending where you are; z+ prints the next screenful. If you want less than a screenful of lines, type z.11 to display five lines before and five lines after the current line. (Typing z.n, when n is an odd number, displays a total of n lines, centered about the current line; when n is an even number, it displays n−1 lines, so that the lines displayed are centered around the current line.) You can give counts after other commands; for example, you can delete 5 lines starting with the current line with the command d5.
To find things in the file, you can use line numbers if you happen to know them; since the line numbers change when you insert and delete lines this is somewhat unreliable. You can search backwards and forwards in the file for strings by giving commands of the form /text/ to search forward for text or ?text? to search backward for text. If a search reaches the end of the file without finding text, it wraps around and continues to search back to the line where you are. A useful feature here is a search of the form /^text/ which searches for text at the beginning of a line. Similarly /text$/ searches for text at the end of a line. You can leave off the trailing / or ? in these commands.

The current line has the symbolic name dot (.); this is most useful in a range of lines as in ., $p which prints the current line plus the rest of the lines in the file. To move to the last line in the file, you can refer to it by its symbolic name $. Thus the command $d deletes the last line in the file, no matter what the current line is. Arithmetic with line references is also possible. Thus the line $ - 5 is the fifth before the last and . + 20 is 20 lines after the current line.

You can find out the current line by typing . = . This is useful if you wish to move or copy a section of text within a file or between files. Find the first and last line numbers you wish to copy or move. To move lines 10 through 20, type 10, 20d a to delete these lines from the file and place them in a buffer named a. edit has 26 such buffers named a through z. To put the contents of buffer a after the current line, type put a. If you want to move or copy these lines to another file, execute an edit (e) command after copying the lines; following the e command with the name of the other file you wish to edit, that is, edit chapter2. To copy lines without deleting them, use yank (y) in place of d. If the text you wish to move or copy is all within one file, it is not necessary to use named buffers. For example, to move lines 10 through 20 to the end of the file, type 10, 20m $.

OPTIONS

These options can be turned on or off using the set command in ex(1).

- l - s
  Suppress all interactive user feedback. This is useful when processing editor scripts.

- l
  Set up for editing LISP programs.

- L
  List the name of all files saved as the result of an editor or system crash.

- R
  Readonly mode; the readonly flag is set, preventing accidental overwriting of the file.

- x filename
  Edit filename after an editor or system crash. (Recover the version of filename that was in the buffer when the crash occurred.)

- t tag
  Edit the file containing the tag and position the editor at its definition.

- v
  Start up in display editing state using vi. You can achieve the same effect by simply typing the vi command itself.
-V  Verbose. When ex commands are read by means of standard input, the input will be echoed to standard error. This may be useful when processing ex commands within shell scripts.

-x  Encryption option; when used, edit simulates the X command of ex and prompts the user for a key. This key is used to encrypt and decrypt text using the algorithm of the crypt command. The X command makes an educated guess to determine whether text read in is encrypted or not. The temporary buffer file is encrypted also, using a transformed version of the key typed in for the -x option.

-wn  Set the default window size to n. This is useful when using the editor over a slow speed line.

-C  Encryption option; same as the -x option, except that vi simulates the C command of ex. The C command is like the X command of ex, except that all text read in is assumed to have been encrypted.

+command  |  -c command  Begin editing by executing the specified editor command (usually a search or positioning command).

The filename argument indicates one or more files to be edited.

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

<table>
<thead>
<tr>
<th>/usr/bin/edit</th>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
<td></td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>/usr/xpg4/bin/edit</th>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWxcu4</td>
<td></td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
<td></td>
</tr>
</tbody>
</table>

SEE ALSO  ed(1), ex(1), vi(1), attributes(5), XPG4(5)

NOTES  The encryption options are provided with the Security Administration Utilities package, which is available only in the United States.

The /usr/xpg4/bin/edit utility is identical to /usr/bin/edit.
egrep(1)

### NAME

egrep – search a file for a pattern using full regular expressions

### SYNOPSIS

```
/usr/bin/egrep [-bchilnv] [-e pattern_list] [-f file] [strings] [file...]  
/usr/xpg4/bin/egrep [-bchilnvx] [-e pattern_list] [-f file] [strings] 
[file...]  
```

### DESCRIPTION

The `egrep` utility searches files for a pattern of characters and prints all lines that contain that pattern. `egrep` uses full regular expressions (expressions that have string values that use the full set of alphanumeric and special characters) to match the patterns. It uses a fast deterministic algorithm that sometimes needs exponential space.

If no files are specified, `egrep` assumes standard input. Normally, each line found is copied to the standard output. The file name is printed before each line found if there is more than one input file.

The `/usr/bin/egrep` utility accepts full regular expressions as described on the `regexp(5)` manual page, except for \{ and \}, \{ and \}, \{ and \}, \< and \>, and \n, and with the addition of:

1. A full regular expression followed by + that matches one or more occurrences of the full regular expression.
2. A full regular expression followed by ? that matches 0 or 1 occurrences of the full regular expression.
3. Full regular expressions separated by | or by a NEWLINE that match strings that are matched by any of the expressions.
4. A full regular expression that may be enclosed in parentheses () for grouping.

Be careful using the characters $, * [, ^, (, ), and \ in full regular expression, because they are also meaningful to the shell. It is safest to enclose the entire full regular expression in single quotes "... ".

The order of precedence of operators is [ ], then * ?, +, then concatenation, then | and NEWLINE.

The `/usr/xpg4/bin/egrep` utility uses the regular expressions described in the EXTENDED REGULAR EXPRESSIONS section of the `regex(5)` manual page.

### OPTIONS

The following options are supported for both `/usr/bin/egrep` and `/usr/xpg4/bin/egrep`:

- `-b` Precede each line by the block number on which it was found. This can be useful in locating block numbers by context (first block is 0).
- `-c` Print only a count of the lines that contain the pattern.
- `-e pattern_list` Search for a `pattern_list` (full regular expression that begins with a `-`).
- `-f file` Take the list of full regular expressions from file.
egrep(1)

- Suppress printing of filenames when searching multiple files.
- Ignore upper/lower case distinction during comparisons.
- Print the names of files with matching lines once, separated by NEWLINEs. Does not repeat the names of files when the pattern is found more than once.
- Precede each line by its line number in the file (first line is 1).
- Work silently, that is, display nothing except error messages. This is useful for checking the error status.
- Print all lines except those that contain the pattern.

The following option is supported for /usr/xpg4/bin/egrep only:

- Consider only input lines that use all characters in the line to match an entire fixed string or regular expression to be matching lines.

The following operands are supported:

- A path name of a file to be searched for the patterns. If no file operands are specified, the standard input will be used.
- Specify a pattern to be used during the search for input.
- Specify one or more patterns to be used during the search for input. This operand is treated as if it were specified as -pattern_list.

See largefile(5) for the description of the behavior of egrep when encountering files greater than or equal to 2 Gbyte (2^31 bytes).

See environ(5) for descriptions of the following environment variables that affect the execution of egrep: LC_COLLATE, LC_CTYPE, LC_MESSAGES, and NLSPATH.

The following exit values are returned:

0 If any matches are found.
1 If no matches are found.
2 For syntax errors or inaccessible files (even if matches were found).

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

<table>
<thead>
<tr>
<th>/usr/bin/egrep</th>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
<td></td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
<td></td>
</tr>
</tbody>
</table>
egrep(1)
/usr/xpg4/bin/egrep

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWxcu4</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>


NOTES  Ideally there should be only one grep command, but there is not a single algorithm that spans a wide enough range of space-time tradeoffs.

Lines are limited only by the size of the available virtual memory.

The /usr/xpg4/bin/egrep utility is identical to /usr/xpg4/bin/grep -E (see grep(1)). Portable applications should use /usr/xpg4/bin/grep -E.
**NAME**

`eject` – eject media such as CD-ROM and floppy from drive

**SYNOPSIS**

```
eject [-dfnpq] [device | nickname]
```

**DESCRIPTION**

The `eject` utility is used for those removable media devices that do not have a manual eject button, or for those that do, but are managed by Volume Management (see `vold(1M)`). The device may be specified by its name or by a nickname; if Volume Management is running and no device is specified, the default device is used.

Only devices that support `eject` under program control respond to this command. `eject` responds differently, depending on whether or not Volume Management is running.

**With Volume Management**

When `eject` is used on media that can only be ejected manually, it will do everything except remove the media, including unmounting the file system if it is mounted. In this case, `eject` displays a message that the media can now be manually ejected. If a window system is running, the message is displayed as a pop-up window, unless the `-p` option is supplied. If no window system is running or the `-p` option is supplied, a message is displayed both to `stderr` and to the system console that the media can now be physically removed.

Volume Management has the concept of a default device, which `eject` uses if no pathname or nickname is specified. Use the `-d` option to check what default device will be used.

**Without Volume Management**

When Volume Management is not running and a pathname is specified, `eject` sends the eject command to that pathname. If a nickname is supplied instead of a pathname, `eject` will recognize the following list:

<table>
<thead>
<tr>
<th>Nickname</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>fd</td>
<td>/dev/rdiskette</td>
</tr>
<tr>
<td>fd0</td>
<td>/dev/rdiskette</td>
</tr>
<tr>
<td>fd1</td>
<td>/dev/rdiskette1</td>
</tr>
<tr>
<td>diskette</td>
<td>/dev/rdiskette</td>
</tr>
<tr>
<td>diskette0</td>
<td>/dev/rdiskette0</td>
</tr>
<tr>
<td>diskette1</td>
<td>/dev/rdiskette1</td>
</tr>
<tr>
<td>rdiskette</td>
<td>/dev/rdiskette</td>
</tr>
<tr>
<td>rdiskette0</td>
<td>/dev/rdiskette0</td>
</tr>
<tr>
<td>rdiskette1</td>
<td>/dev/rdiskette1</td>
</tr>
<tr>
<td>floppy</td>
<td>/dev/rdiskette</td>
</tr>
<tr>
<td>floppy0</td>
<td>/dev/rdiskette0</td>
</tr>
</tbody>
</table>
The list above can be reproduced with the -n option.

Do not physically eject media from a device which contains mounted file systems. eject automatically searches for any mounted file systems which reside on the device and attempts to umount them prior to ejecting the media (see mount(1M)). If the unmount operation fails, eject prints a warning message and exits. The -f option may be used to specify an eject even if the device contains mounted partitions; this option works only if Volume Management is not running.

eject can also display its default device and a list of nicknames.

If you have inserted a floppy diskette, you must use volcheck(1) before ejecting the media to inform Volume Management of the floppy's presence.

OPTIONS
The following options are supported:

- \( -d \) Displays the name of the default device to be ejected.
- \( -f \) Forces the device to eject even if it is busy, if Volume Management is not running.
- \( -n \) Displays the nickname to device name translation table.
- \( -p \) Does not try to call the eject_popup program.
- \( -q \) Queries to see if the media is present.

OPERANDS
The following operands are supported:

device Specifies which device to eject, by the name it appears in the directory /dev.

nickname Specifies which device to eject, by its nickname as known to this command.

EXAMPLES

**EXAMPLE 1** Ejecting a CD while Volume Management is running

To eject a CD from its drive, while Volume Management is running (assuming only one CD-ROM drive):

```
example> eject cdrom0
```

**EXAMPLE 2** Ejecting a CD-ROM without running Volume Management

To eject a CD-ROM drive with pathname /dev/dsk/c0t3d0s2, without Volume Management running:

```
example> eject /dev/dsk/c0t3d0s2
```
EXAMPLE 3  Ejecting a floppy disk

To eject a floppy disk (whether or not Volume Management is running):

example> eject floppy0

EXIT STATUS

The following exit codes are returned:

0  The operation was successful or, with the -q option, the media is in the drive.

1  The operation was unsuccessful or, with the -q option, the media is not in the drive.

2  Invalid options were specified.

3  An ioctl() request failed.

4  Manually ejectable media is now okay to remove.

FILES

/dev/diskette0    default diskette file
/dev/sr0         default CD-ROM file (deprecated)
/dev/dsk/c0t6d0s2 default CD-ROM file
/usr/lib/vold/eject_popup popup used for manually ejected media

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO

volcancel(1), volcheck(1), volmissing(1), mount(1M), rmmount(1M), vold(1M), ioctl(2), rmmount.conf(4), vold.conf(4), attributes(5), volfs(7FS)

DIAGNOSTICS

A short help message is printed if an unknown option is specified. A diagnostic is printed if the device name cannot be opened or does not support eject.

Device Busy  An attempt was made to eject a device that has a mounted file system. A warning message is printed when doing a forced eject of a mounted device.

BUGS

There should be a way to change the default on a per-user basis.

If Volume Management is not running, it is possible to eject a volume that is currently mounted (see mount(1M)). For example, if you have a CD-ROM drive at /dev/dsk/c0t3d0s2 mounted on /mnt, the following command (without Volume Management running) will work:

example> eject /dev/dsk/c0t3d0s0
since both slices s0 and s2 reference the whole CD-ROM drive.
elfdump – dump selected parts of an object file

SYNOPSIS  
elfdump [-CcdeGghikmnprsvy] [-N name] [-w file] filename...

DESCRIPTION  
The elfdump utility symbolically dumps selected parts of the specified object file(s).
The options allow specific portions of the file to be displayed.

The elfdump utility is similar in function to the dump(1) utility, which offers an older and
less user-friendly interface than elfdump, although dump may be more appropriate for certain uses such as in shell scripts.

Archive files, produced by ar(1), can also be inspected with elfdump. In this case each object within the archive is processed using the options supplied.

For a complete description of the displayed information, refer to the Linker and Libraries Guide.

OPTIONS  
The following options are supported:

- `-c` Dumps section header information.
- `-C` Demangles C++ symbol names.
- `-d` Dumps the contents of the .dynamic section.
- `-e` Dumps the ELF header.
- `-g` Dumps the contents of the .group section.
- `-G` Dumps the contents of the .got section.
- `-h` Dumps the contents of the .hash section.
- `-i` Dumps the contents of the .interp section.
- `-k` Calculates the ELF checksum (see gelf_checksum(3ELF)).
- `-m` Dumps the contents of the .SUNW_move section.
- `-n` Dumps the contents of the .note section.
- `-N name` Qualifies an option with a specific name. For example, in a file that contains more than one symbol table, the .dynsym table can be displayed using:
  
  example% elfdump -s -N .dynsym filename

- `-p` Dumps the program headers.
- `-r` Dumps the contents of the relocation sections (that is, .rel[a]).
- `-s` Dumps the contents of the symbol table sections (that is, .dynsym and/or .symtab) and, in the case of archives, dumps the archive symbol table. Individual sections can be specified with the -N option, or an archive symbol table can be specified using the special section name -N ARSYM.
In addition to the standard symbol table information, the version definition index of the symbol is also provided under the `ver` heading.

- **v**
  Dumps the contents of the version sections (that is, `.SUNW_version`).

- **w file**
  Writes the contents of a section specified with the `-N` option to the named file. This is useful for extracting an individual section’s data for additional processing. For example, extracting the `.text` section of a file can be carried out with:

  ```
  example% elfdump -w text.out -N .text filename
  ```

- **γ**
  Dumps the contents of the `.SUNW_syminfo` section.

**OPERANDS**

The following operand is supported:

- **filename**
  The name of the specified object file.

**FILES**

- **liblddbg.so**
  Linker debugging library

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWbtool</td>
</tr>
</tbody>
</table>

**SEE ALSO**

- `ar(1)`, `dump(1)`, `nm(1)`, `pvs(1)`, `elf(3ELF)`, `attributes(5)`
- *Linker and Libraries Guide*
enable(1)

NAME
enable, disable – enable/disable LP printers

SYNOPSIS
/usr/bin/enable printer...
/usr/bin/disable [-c | -W] [-r [reason]] printer...

DESCRIPTION
The enable command activates printers, enabling them to print requests submitted by the \texttt{lp} command. \texttt{enable} must be run on the printer server.

The disable command deactivates printers, disabling them from printing requests submitted by the \texttt{lp} command. By default, any requests that are currently printing on \texttt{printer} will be reprinted in their entirety either on \texttt{printer} or another member of the same class of printers. The disable command must be run on the print server.

Use \texttt{lpstat -p} to check the status of printers.

enable and disable only effect queueing on the print server’s spooling system. Executing these commands from a client system will have no effect on the server.

OPTIONS
The following options are supported for use with disable:

\texttt{-c}  Cancels any requests that are currently printing on \texttt{printer}. This option cannot be used with the \texttt{-W} option. If the printer is remote, the \texttt{-c} option will be silently ignored.

\texttt{-W}  Waits until the request currently being printed is finished before disabling \texttt{printer}. This option cannot be used with the \texttt{-c} option. If the printer is remote, the \texttt{-W} option will be silently ignored.

\texttt{-r [reason]}  Assigns a \textit{reason} for the disabling of the printer(s). This \textit{reason} applies to all printers specified. This \textit{reason} is reported by \texttt{lpstat -p}. Enclose \textit{reason} in quotes if it contains blanks. The default reason is "unknown reason" for the existing printer, and "new printer" for a printer added to the system but not yet enabled.

OPERANDS
The following operand is supported for both enable and disable:

\textit{printer}  The name of the printer to be enabled or disabled. Specify \textit{printer} using atomic name. See printers.conf(4) for information regarding the naming conventions for atomic names.

EXIT STATUS
The following exit values are returned:

0  Successful completion.
non-zero  An error occurred.

FILES
/var/spool/lp/*  LP print queue.

ATTRIBUTES
See attributes(5) for descriptions of the following attributes:
enable(1)

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWpcu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

SEE ALSO  lp(1), lpstat(1), printers.conf(4), attributes(5)
### NAME
env – set environment for command invocation

### SYNOPSIS
/usr/bin/env [-i | -] [name=value...] [utility [arg...]]

/usr/xpg4/bin/env [-i | -] [name=value...] [utility [arg...]]

### DESCRIPTION
The env utility obtains the current environment, modifies it according to its arguments, then invokes the utility named by the utility operand with the modified environment.

Optional arguments are passed to utility. If no utility operand is specified, the resulting environment is written to the standard output, with one name=value pair per line.

If env executes commands with arguments, it uses the default shell /usr/bin/sh (see sh(1)).

If env executes commands with arguments, it uses /usr/xpg4/bin/sh (see ksh(1)).

### OPTIONS
The following options are supported:

- **-i ||** Ignores the environment that would otherwise be inherited from the current shell. Restricts the environment for utility to that specified by the arguments.

### OPERANDS
The following operands are supported:

- **name=value** Arguments of the form name=value modify the execution environment, and are placed into the inherited environment before utility is invoked.

- **utility** The name of the utility to be invoked. If utility names any of the special shell built-in utilities, the results are undefined.

- **arg** A string to pass as an argument for the invoked utility.

### EXAMPLES
**EXAMPLE 1 Invoking utilities with new PATH values**

The following utility:

```bash
example env -i PATH=/mybin mygrep xyz myfile
```

invokes the utility mygrep with a new PATH value as the only entry in its environment. In this case, PATH is used to locate mygrep, which then must reside in /mybin.

### ENVIRONMENT VARIABLES
See environ(5) for descriptions of the following environment variables that affect the execution of env: LC_CTYPE, LC_MESSAGES, and NLSPATH.

### EXIT STATUS
If utility is invoked, the exit status of env is the exit status of utility; otherwise, the env utility is with one of the following values:

- **0** Successful completion.
An error occurred.
utility was found but could not be invoked.
utility could not be found.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWxcu4</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

SEE ALSO ksh(1), sh(1), exec(2), profile(4), attributes(5), environ(5), XPG4(5)
eqn, neqn, checkeq – typeset mathematics test

SYNOPSIS

```
eqn [-d xy] [-f n] [-p n] [-s n] [file...]
neqn [file...]
checkeq [file...]
```

DESCRIPTION

eqn and neqn are language processors to assist in describing equations. eqn is a preprocessor for troff(1) and is intended for devices that can print troff’s output. neqn is a preprocessor for nroff(1) and is intended for use with terminals. Usage is almost always:

```
example% eqn file ... | troff
example% neqn file ... | nroff
```

If no files are specified, eqn and neqn read from the standard input. A line beginning with .EQ marks the start of an equation. The end of an equation is marked by a line beginning with .EN. Neither of these lines is altered, so they may be defined in macro packages to get centering, numbering, and so on. It is also possible to set two characters as “delimiters”; subsequent text between delimiters is also treated as eqn input.

checkeq reports missing or unbalanced delimiters and .EQ/.EN pairs.

OPTIONS

The following options are supported:

```
-d xy      Sets equation delimiters set to characters x and y with the command-line argument. The more common way to do this is with delim xy between .EQ and .EN. The left and right delimiters may be identical. Delimiters are turned off by delimoff appearing in the text. All text that is neither between delimiters nor between .EQ and .EN is passed through untouched.
-f n       Changes font to n globally in the document. The font can also be changed globally in the body of the document by using the gfont n directive, where n is the font specification.
-p n       Reduces subscripts and superscripts by n point sizes from the previous size. In the absence of the -p option, subscripts and superscripts are reduced by 3 point sizes from the previous size.
-s n       Changes point size to n globally in the document. The point size can also be changed globally in the body of the document by using the gsize n directive, where n is the point size.
```

OPERANDS

The following operands are supported:

```
file       The nroff or troff file processed by eqn or neqn.
```
The nroff version of this description depicts the output of neqn to the terminal screen exactly as neqn is able to display it. To see an accurate depiction of the output, view the printed version of this page.

Tokens within eqn are separated by braces, double quotes, tildes, circumflexes, SPACE, TAB, or NEWLINE characters. Braces { } are used for grouping. Generally speaking, anywhere a single character like x could appear, a complicated construction enclosed in braces may be used instead. A tilde (~) represents a full SPACE in the output; a circumflex (^) half as much.

Subscripts and superscripts:
These are produced with the keywords sub and sup.

\[ x \text{ sub i} \quad \text{makes } x_i \]
\[ a \text{ sub i sup 2} \quad \text{produces } a_i^2 \]
\[ e \text{ sup \{x sup 2 + y sup 2\}} \quad \text{gives } e^{x^2+y^2} \]

Fractions:
Fractions are made with over.

\[ \frac{a}{b} \quad \text{yields} \]
\[ \frac{a}{b} \]

Square Roots:
These are made with sqrt

\[ \frac{1}{\sqrt{ax^2+bx+c}} \quad \text{results in} \]
\[ \frac{1}{\sqrt{ax^2+bx+c}} \]

Limits:
The keywords from and to introduce lower and upper limits on arbitrary things:

\[ \lim_{n \to \infty} \sum_{0}^{n} x \text{ sub i} \quad \text{makes} \]
Brackets and Braces:

Left and right brackets, braces, etc., of the right height are made with \texttt{left} and \texttt{right}.

\texttt{left [ x \sup 2 + y \sup 2 over alpha right ]} \rightarrow -1

produces

\[
\left[ \frac{x^2 + y^2}{\alpha} \right] = 1
\]

The right clause is optional. Legal characters after \texttt{left} and \texttt{right} are braces, brackets, bars, c and f for ceiling and floor, and " " for nothing at all (useful for a right-side-only bracket).

Vertical piles:

Vertical piles of things are made with \texttt{pile}, \texttt{lpile}, \texttt{cpile}, and \texttt{rpile}.

\texttt{pile \{ a above b above c \}}

produces

\[
a
\]
\[
b
\]
\[
c
\]

There can be an arbitrary number of elements in a pile. \texttt{lpile} left-justifies, \texttt{pile} and \texttt{cpile} center, with different vertical spacing, and \texttt{rpile} right justifies.

Matrices:

Matrices are made with \texttt{matrix}.

\texttt{matrix \{ lcol \{ x sub i above y sub 2 \} ccol \{ 1 above 2 \} \}}

produces

\[
x_i \quad 1
\]
\[
y_2 \quad 2
\]

In addition, there is \texttt{rcol} for a right-justified column.
Diacritical marks:
Diacritical marks are made with dot, dotdot, hat, tilde, bar, vec, dyad, and under.

\[ x \, \text{dot} = f(t) \, \text{bar} \]
\[ \dot{x} = f(t) \]

\[ y \, \text{dotdot} \, \text{bar} = n \, \text{under} \]
\[ \ddot{y} = n \]

\[ x \, \text{vec} = y \, \text{dyad} \]
\[ \vec{x} = \hat{y} \]

Sizes and Fonts:
Sizes and font can be changed with size \( n \) or size \( \pm n \), roman, italic, bold, and font \( n \). Size and fonts can be changed globally in a document by gsize \( n \) and gfont \( n \), or by the command-line arguments -sn and -fn.

Successive display arguments:
Successive display arguments can be lined up. Place mark before the desired lineup point in the first equation; place lineup at the place that is to line up vertically in subsequent equations.

Shorthands:
Shorthands may be defined or existing keywords redefined with define:

\[ \text{define } \text{thing } \% \text{replacement } \% \]
\[ \text{Defines a new token called thing which will be replaced by replacement whenever it appears thereafter. The } \% \text{ may be any character that does not occur in replacement.} \]

Keywords and Shorthands:
Keywords like sum int inf and shorthands like >= \rightarrow and != are recognized.

Greek letters:
Greek letters are spelled out in the desired case, as in alpha or GAMMA.
Mathematical words:

Mathematical words like \textit{sin}, \textit{cos}, and \textit{log} are made Roman automatically.

\texttt{troff(1)} four-character escapes like \texttt{	extbackslash bu (•)} can be used anywhere. Strings enclosed in double quotes "..." are passed through untouched; this permits keywords to be entered as text, and can be used to communicate with \texttt{troff} when all else fails.

\textbf{ATTRIBUTES} See attributes(5) for descriptions of the following attributes:

\begin{center}
\begin{tabular}{|l|l|}
\hline
\textbf{ATTRIBUTE TYPE} & \textbf{ATTRIBUTE VALUE} \\
\hline
Availability & SUNWdoc \\
\hline
\end{tabular}
\end{center}

\textbf{SEE ALSO} \texttt{nroff(1), tbl(1), troff(1), attributes(5), ms(5)}

\textbf{BUGS} To embolden characters such as digits and parentheses, it is necessary to quote them, as in \texttt{\textbf{bold "12.3"}}.
error(1)

## NAME

error – insert compiler error messages at right source lines

## SYNOPSIS

```
error [-n] [-q] [-s] [-v] [-t suffixlist] [-I ignorefile] [filename]
```

## DESCRIPTION

error analyzes error messages produced by a number of compilers and language processors. It replaces the painful, traditional methods of scribbling abbreviations of errors on paper, and permits error messages and source code to be viewed simultaneously.

error looks at error messages, either from the specified file `filename` or from the standard input, and:

- Determines which language processor produced each error message.
- Determines the file name and line number of the erroneous line.
- Inserts the error message into the source file immediately preceding the erroneous line.

Error messages that can’t be categorized by language processor or content are not inserted into any file, but are sent to the standard output. error touches source files only after all input has been read.

error is intended to be run with its standard input connected with a pipe to the error message source. Some language processors put error messages on their standard error file; others put their messages on the standard output. Hence, both error sources should be piped together into error. For example, when using the csh syntax, the following command analyzes all the error messages produced by whatever programs make(1S) runs when making lint:

```
eexample% make -s lint | & error -q -v
```

error knows about the error messages produced by: as(1), cpp(1), ld(1), cc(1B), make(1S) and other compilers. For all languages except Pascal, error messages are restricted to one line. Some error messages refer to more than one line in more than one file, in which case error duplicates the error message and inserts it in all the appropriate places.

## OPTIONS

- `-n`  
  Do not touch any files; all error messages are sent to the standard output.

- `-q`  
  error asks whether the file should be touched. A ‘y’ or ‘n’ to the question is necessary to continue. Absence of the `-q` option implies that all referenced files (except those referring to discarded error messages) are to be touched.

- `-s`  
  Print out statistics regarding the error categorization.

- `-v`  
  After all files have been touched, overlay the visual editor `vi` with it set up to edit all files touched, and positioned in the first touched file at the first error. If `vi` can’t be found, try `ex` or `ed` from standard places.
error(1)

-t suffixlist  Take the following argument as a suffix list. Files whose suffixes do not appear in the suffix list are not touched. The suffix list is dot separated, and ‘*’ wildcards work. Thus the suffix list:

  .c.y.f*.h

allows error to touch files ending with ‘.c’, ‘.y’, ‘.f*’ and ‘.h’.

error catches interrupt and terminate signals, and terminates in an orderly fashion.

EXAMPLES

**EXAMPLE 1** Examples of the error command.

In the following C shell (/usr/bin/csh) example, error takes its input from the FORTRAN compiler:

```
example% f77 -c any.f |& error options
```

Here is the same example using the Korn shell (/usr/bin/ksh):

```
example% f77 -c any.f 2>&1 | error options
```

**USAGE** error does one of six things with error messages.

**synchronize**  Some language processors produce short errors describing which file they are processing. error uses these to determine the file name for languages that do not include the file name in each error message. These synchronization messages are consumed entirely by error.

**discard**  Error messages from lint that refer to one of the two lint libraries, /usr/lib/lint/llib-lc and /usr/lib/lint/llib-port are discarded, to prevent accidentally touching these libraries. Again, these error messages are consumed entirely by error.

**nullify**  Error messages from lint can be nullified if they refer to a specific function, which is known to generate diagnostics which are not interesting. Nullified error messages are not inserted into the source file, but are written to the standard output. The names of functions to ignore are taken from either the file named .errorrc in the user’s home directory, or from the file named by the -I option. If the file does not exist, no error messages are nullified. If the file does exist, there must be one function name per line.
error(1)

| not file specific | Error messages that can’t be intuited are grouped together, and written to the standard output before any files are touched. They are not inserted into any source file. |
| file specific     | Error messages that refer to a specific file but to no specific line are written to the standard output when that file is touched. |
| true errors       | Error messages that can be intuited are candidates for insertion into the file to which they refer. |

Only true error messages are inserted into source files. Other error messages are consumed entirely by `error` or are written to the standard output. `error` inserts the error messages into the source file on the line preceding the line number in the error message. Each error message is turned into a one line comment for the language, and is internally flagged with the string `###` at the beginning of the error, and `%%` at the end of the error. This makes pattern searching for errors easier with an editor, and allows the messages to be easily removed. In addition, each error message contains the source line number for the line the message refers to. A reasonably formatted source program can be recompiled with the error messages still in it, without having the error messages themselves cause future errors. For poorly formatted source programs in free format languages, such as C or Pascal, it is possible to insert a comment into another comment, which can wreak havoc with a future compilation. To avoid this, format the source program so there are no language statements on the same line as the end of a comment.

FILES

- `~/.errorrc`  function names to ignore for lint error messages
- `/dev/tty`  user’s teletype

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWbtool</td>
</tr>
</tbody>
</table>

SEE ALSO

as(1), cc(1B), cpp(1), csh(1), ed(1), ex(1), make(1S), ld(1), vi(1), attributes(5)

BUGS

Opens the tty-device directly for user input.

Source files with links make a new copy of the file with only one link to it.

Changing a language processor’s error message format may cause `error` to not understand the error message.

`error`, since it is purely mechanical, will not filter out subsequent errors caused by “floodgating” initiated by one syntactically trivial error. Humans are still much better at discarding these related errors.

348  man pages section 1: User Commands • Last Revised 5 Mar 1992
Pascal error messages belong after the lines affected, error puts them before. The alignment of the `|` marking the point of error is also disturbed by error.

error was designed for work on CRT 's at reasonably high speed. It is less pleasant on slow speed terminals, and was not designed for use on hardcopy terminals.
The `ex` utility is the root of a family of editors: `ex` and `vi`. `ex` is a superset of `ed(1)`, with the most notable extension being a display editing facility. Display based editing is the focus of `vi).

If you have a CRT terminal, you may wish to use a display based editor; in this case see `vi(1)`, which is a command which focuses on the display-editing portion of `ex`.

If you have used `ed` you will find that, in addition to having all of the `ed` commands available, `ex` has a number of additional features useful on CRT terminals. Intelligent terminals and high speed terminals are very pleasant to use with `vi`. Generally, the `ex` editor uses far more of the capabilities of terminals than `ed` does, and uses the terminal capability database (see `terminfo(4)`) and the type of the terminal you are using from the environment variable `TERM` to determine how to drive your terminal efficiently. The editor makes use of features such as insert and delete character and line in its `visual` command (which can be abbreviated `vi`) and which is the central mode of editing when using the `vi` command.

The `ex` utility contains a number of features for easily viewing the text of the file. The `z` command gives easy access to windows of text. Typing `CTRL-D` causes the editor to scroll a half-window of text and is more useful for quickly stepping through a file than just typing return. Of course, the screen-oriented `visual` mode gives constant access to editing context.

The `ex` utility gives you help when you make mistakes. The `undo` command allows you to reverse any single change which goes astray. `ex` gives you a lot of feedback, normally printing changed lines, and indicates when more than a few lines are affected by a command so that it is easy to detect when a command has affected more lines than it should have.

The editor also normally prevents overwriting existing files, unless you edited them, so that you do not accidentally overwrite a file other than the one you are editing. If the system (or editor) crashes, or you accidentally hang up the telephone, you can use the editor `recover` command (or `-r file` option) to retrieve your work. This will get you back to within a few lines of where you left off.

The `ex` utility has several features for dealing with more than one file at a time. You can give it a list of files on the command line and use the `next` command to deal with each in turn. The `next` command can also be given a list of file names, or a pattern as used by the shell to specify a new set of files to be dealt with. In general, file names in the editor may be formed with full shell metasyntax. The metacharacter `%' is also available in forming file names and is replaced by the name of the current file.
The editor has a group of buffers whose names are the ASCII lower-case letters (a-z). You can place text in these named buffers where it is available to be inserted elsewhere in the file. The contents of these buffers remain available when you begin editing a new file using the edit (e) command.

There is a command & in ex which repeats the last substitute command. In addition, there is a confirmed substitute command. You give a range of substitutions to be done and the editor interactively asks whether each substitution is desired.

It is possible to ignore the case of letters in searches and substitutions. ex also allows regular expressions which match words to be constructed. This is convenient, for example, in searching for the word “edit” if your document also contains the word “editor.”

ex has a set of options which you can set to tailor it to your liking. One option which is very useful is the autoindent option that allows the editor to supply leading white space to align text automatically. You can then use ^D as a backtab and space or tab to move forward to align new code easily.

Miscellaneous useful features include an intelligent join (j) command that supplies white space between joined lines automatically, commands < and > which shift groups of lines, and the ability to filter portions of the buffer through commands such as sort.

OPTIONS

The following options are supported:

- `-l -s` Suppress all interactive user feedback. This is useful when processing editor scripts.
- `-l` Set up for editing LISP programs.
- `-L` List the name of all files saved as the result of an editor or system crash.
- `-R` Readonly mode; the readonly flag is set, preventing accidental overwriting of the file.
- `-r file` Edit file after an editor or system crash. (Recovers the version of file that was in the buffer when the crash occurred.)
- `-t tag` Edit the file containing the tag and position the editor at its definition.
- `-v` Start up in display editing state using vi. You can achieve the same effect by simply typing the vi command itself.
- `-V` Verbose. When ex commands are read by means of standard input, the input will be echoed to standard error. This may be useful when processing ex commands within shell scripts.
Encryption option. Simulates the `x` command and prompts the user for a key. This key is used to encrypt and decrypt text using the algorithm of the `crypt` command. The `x` command makes an educated guess to determine whether text read in is encrypted or not. The temporary buffer file is encrypted also, using a transformed version of the key typed in for the `-x` option.

`-wn` Set the default window size to `n`. This is useful when using the editor over a slow speed line.

`-c` Encryption option. Same as the `-x` option, except simulates the `c` command. The `c` command is like the `x` command, except that all text read in is assumed to have been encrypted.

`+command | -c command` Begin editing by executing the specified editor command (usually a search or positioning command).

If both the `-t tag` and the `-c command` options are given, the `-t tag` will be processed first. That is, the file containing the tag is selected by `-t` and then the command is executed.

**OPERANDS**

The following operand is supported:

| file | A path name of a file to be edited. |

**ex States**

Command Normal and initial state. Input prompted for by “:”. Your line kill character cancels a partial command.

Insert Entered by `a`, `i`, or `c`. Arbitrary text may be entered. Insert state normally is terminated by a line having only “.” on it, or, abnormally, with an interrupt.

Visual Entered by typing `vi`; terminated by typing `q` or `^\` (CTRL-\).

**ex Command Names and Abbreviations**

<table>
<thead>
<tr>
<th>Command Name</th>
<th>Abbrev-iation</th>
<th>Command Name</th>
<th>Abbrev-iation</th>
<th>Command Name</th>
<th>Abbrev-iation</th>
</tr>
</thead>
<tbody>
<tr>
<td>abbrev</td>
<td>ab</td>
<td>map</td>
<td>set</td>
<td>se</td>
<td></td>
</tr>
<tr>
<td>append</td>
<td>a</td>
<td>mark</td>
<td>ma</td>
<td>shell</td>
<td>sh</td>
</tr>
<tr>
<td>args</td>
<td>ar</td>
<td>move</td>
<td>m</td>
<td>source</td>
<td>so</td>
</tr>
<tr>
<td>change</td>
<td>c</td>
<td>next</td>
<td>n</td>
<td>substitute</td>
<td>s</td>
</tr>
<tr>
<td>copy</td>
<td>co</td>
<td>number</td>
<td>nu</td>
<td>unabbrev</td>
<td>unab</td>
</tr>
<tr>
<td>delete</td>
<td>d</td>
<td>preserve</td>
<td>pre</td>
<td>undo</td>
<td>u</td>
</tr>
<tr>
<td>edit</td>
<td>e</td>
<td>print</td>
<td>p</td>
<td>unmap</td>
<td>unm</td>
</tr>
</tbody>
</table>
Command Name | Abbrev- iation | Command Name | Abbrev- iation | Command Name | Abbrev- iation
--- | --- | --- | --- | --- | ---
file | f | put | pu | version | ve
global | g | quit | q | visual | vi
insert | i | read | r | write | w
join | j | recover | rec | xit | x
list | l | rewind | rew | yank | ya

For all of the `ex` commands listed below, if both a count and a range are specified for a command that uses them, the number of lines affected will be taken from the count value rather than the range. The starting line for the command is taken to be the first line addressed by the range.

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abbreviate</td>
<td>ab[brev] word rhs</td>
</tr>
<tr>
<td>Append</td>
<td>[line] a[ppend]!</td>
</tr>
<tr>
<td>Arguments</td>
<td>ar[gs]</td>
</tr>
<tr>
<td>Change</td>
<td>[range] c[hange]! [count]</td>
</tr>
<tr>
<td>Change Directory</td>
<td>chd[ir]! [directory]; cd[!] [directory]</td>
</tr>
<tr>
<td>Copy</td>
<td>[range] c[o]py line [flags]; [range] t line [flags]</td>
</tr>
<tr>
<td>Delete</td>
<td>[range] d[ele]te [buffer] [count] [flags]</td>
</tr>
<tr>
<td>Edit</td>
<td>e[dit]! [+line][file]; ex[!] [+line] [file]</td>
</tr>
<tr>
<td>File</td>
<td>f[ile] [file]</td>
</tr>
<tr>
<td>Global</td>
<td>[range] g[lobal] /pattern/ [commands]; [range] v /pattern/ [commands]</td>
</tr>
<tr>
<td>Insert</td>
<td>[line] i[ntsert]!</td>
</tr>
<tr>
<td>Join</td>
<td>[range] j[o]in! [count] [flags]</td>
</tr>
<tr>
<td>List</td>
<td>[range] l[ist] [count] [flags]</td>
</tr>
<tr>
<td>Map</td>
<td>map[!] [x rhs]</td>
</tr>
<tr>
<td>Mark</td>
<td>[line] ma[rk] x; [line] k x</td>
</tr>
<tr>
<td>Move</td>
<td>[range] m[ove] line</td>
</tr>
<tr>
<td>Next</td>
<td>n[ext]! [file ...]</td>
</tr>
<tr>
<td>Number</td>
<td>[range] n[u]mer [count] [flags]; [range] # [count] [flags]</td>
</tr>
<tr>
<td>Open</td>
<td>[line] o[pen] /pattern/ [flags]</td>
</tr>
<tr>
<td>Preserve</td>
<td>pre[serve]</td>
</tr>
</tbody>
</table>
Print  [range] p[rint] [count] [flags]
Put    [line] pu[t] [buffer]
Quit   q[uit]![]
Read   [line] r[ead]![ ] [file]
Recover rec[over] file
Rewind rew[ind]![ ] Set se[t] [option=[value]... ] [nooption... ] [option?]... [all]
Shell  sh[ell]
Source so[urce] file
Substitute [range] s[ubstitute] [/pattern/repl/[options] [count] [flags]]
Suspend su[spend]![ ]; st[op]![ ]
Tag    ta[g]![ ] tagstring
Unabbreviate una[bbrev] word
Undo   u[ndo]
Unmap  unm[ap]! x
Visual [line] vi[ual] [type] [count] [flags]
Write  [range] w[rite]! [>>] [file]; [range] w[rite] ![ ] [file];
       [range] wq![ ] [>>] [file]
Write and Exit [range] x[it]![ ] [file]
Yank   [range] ya[nk] [buffer] [count]
Adjust Window [line] z [type] [count] [flags]
Escape  ! command [range]! command
Shift Left [range] < [count] [flags]
Shift Right [range] > [count] [flags]
Resubstitute [range] & [options] [count] [flags]; [range] s[ubstitute]
          [options] [count] [flags]; [range] ~ [options] [count]
          [flags]
Scroll  EOF
Write Line Number [line] = [flags]
Execute   @ buffer; * buffer

**ex Commands**

C     forced encryption
X     heuristic encryption

354  man pages section 1: User Commands • Last Revised 18 Mar 1997
&       resubst
CR     print next
>      rshift
<      lshift
^D     scroll
z      window
!      shell escape
n      line n
.      current
$      last
+      next
−      previous
+n     n forward
%      1,$
/pat   next with pat
?pat   previous with pat
x=n    n before x
x,y    x through n
'x     marked with x
"      previous context

ex Command Addresses

Initializing options

EXINIT           place set's here in environment variable
$HOME/.exrc      editor initialization file
./.exrc          editor initialization file
set x            enable option x
set nox           disable option x
set x=val         give value val to option x
set               show changed options
set all           show all options
set x?            show value of option x

User Commands  355
**ex(1)**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>autowrite</td>
<td>aw write before changing files</td>
</tr>
<tr>
<td>directory</td>
<td>ex pathname of directory for temporary work files</td>
</tr>
<tr>
<td>exrc</td>
<td>ex allow vi/ex to read the .exrc in the current directory. This option is set in the EXINIT shell variable or in the .exrc file in the $HOME directory.</td>
</tr>
<tr>
<td>ignorecase</td>
<td>ic ignore case of letters in scanning</td>
</tr>
<tr>
<td>list</td>
<td>print ^I for tab, $ at end</td>
</tr>
<tr>
<td>magic</td>
<td>treat [ * special in patterns</td>
</tr>
<tr>
<td>modelines</td>
<td>first five lines and last five lines executed as vi/ex commands if they are of the form ex:command: or vi:command:</td>
</tr>
<tr>
<td>number</td>
<td>nu number lines</td>
</tr>
<tr>
<td>paragraphs</td>
<td>para macro names that start paragraphs</td>
</tr>
<tr>
<td>redraw</td>
<td>sect simulate smart terminal</td>
</tr>
<tr>
<td>report</td>
<td>shows command mode lines</td>
</tr>
<tr>
<td>scroll</td>
<td>sect macro names that start sections</td>
</tr>
<tr>
<td>shiftwidth</td>
<td>sw for &lt; &gt;, and input ^D</td>
</tr>
<tr>
<td>showmatch</td>
<td>sm to ) and } as typed</td>
</tr>
<tr>
<td>showmode</td>
<td>smd show insert mode in vi</td>
</tr>
<tr>
<td>slowopen</td>
<td>slow stop updates during insert</td>
</tr>
<tr>
<td>term</td>
<td>specifies to vi the type of terminal being used (the default is the value of the environment variable TERM)</td>
</tr>
<tr>
<td>window</td>
<td>wm visual mode lines</td>
</tr>
<tr>
<td>wrapmargin</td>
<td>wm automatic line splitting</td>
</tr>
<tr>
<td>wrapscan</td>
<td>ws search around end (or beginning) of buffer</td>
</tr>
</tbody>
</table>

**Scanning pattern formation**

- ^   beginning of line
- $   end of line
- .   any character
- \<  beginning of word
- \>  end of word
| [str] | any character in str |
|[^str] | any character not in str |
|[xy] | any character between x and y |
|* | any number of preceding characters |

**ENVIRONMENT VARIABLES**

See `environ(5)` for descriptions of the following environment variables that affect the execution of `ex`: HOME, PATH, SHELL, TERM, LC_COLLATE, LC_CTYPE, LC_MESSAGES, and NLSPATH.

- **COLUMNS** Override the system-selected horizontal screen size.
- **EXINIT** Determine a list of ex commands that are executed on editor start-up, before reading the first file. The list can contain multiple commands by separating them using a vertical-line (|) character.
- **LINES** Override the system-selected vertical screen size, used as the number of lines in a screenful and the vertical screen size in visual mode.

**EXIT STATUS**
The following exit values are returned:

- 0  Successful completion.
- >0  An error occurred.

**FILES**

- `/var/tmp/Exnnnnn` editor temporary
- `/var/tmp/Rxnnnnn` named buffer temporary
- `/usr/lib/expreserve` preserve command
- `/usr/lib/exrecover` recover command
- `/usr/lib/exstrings` error messages
- `/usr/share/lib/terminfo/*` describes capabilities of terminals
- `/var/preserve/login` preservation directory (where login is the user’s login)
- `${HOME}/.exrc` editor startup file
- `./.exrc` editor startup file

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>/usr/bin/ex</th>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
<td></td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
<td></td>
</tr>
</tbody>
</table>
### SEE ALSO
ed(1), edit(1), grep(1), sed(1), sort(1), vi(1), curses(3CURSES), term(4), terminfo(4), attributes(5), environ(5), standards(5)

*Solaris Advanced User’s Guide*

### AUTHOR
The vi and ex utilities are based on software developed by The University of California, Berkeley California, Computer Science Division, Department of Electrical Engineering and Computer Science.

### NOTES
Several options, although they continue to be supported, have been replaced in the documentation by options that follow the Command Syntax Standard (see intro(1)).

The `−` option has been replaced by `-s`, a `-r` option that is not followed with an option-argument has been replaced by `-L`, and `+command` has been replaced by `-c` command.

The message `file too large to recover with -r option`, which is seen when a file is loaded, indicates that the file can be edited and saved successfully, but if the editing session is lost, recovery of the file with the `-r` option will not be possible.

The `z` command prints the number of logical rather than physical lines. More than a screen full of output may result if long lines are present.

File input/output errors do not print a name if the command line `-s` option is used.

The editing environment defaults to certain configuration options. When an editing session is initiated, ex attempts to read the EXINIT environment variable. If it exists, the editor uses the values defined in EXINIT, otherwise the values set in `$HOME/.exrc` are used. If `$HOME/.exrc` does not exist, the default values are used.

To use a copy of `.exrc` located in the current directory other than `$HOME`, set the `exrc` option in EXINIT or `$HOME/.exrc`. Options set in EXINIT can be turned off in a local `.exrc` only if `exrc` is set in EXINIT or `$HOME/.exrc`.

There is no easy way to do a single scan ignoring case.

The editor does not warn if text is placed in named buffers and not used before exiting the editor.

Null characters are discarded in input files and cannot appear in resultant files.

The standard Solaris version of `ex` will be replaced by the POSIX.2-conforming version (see standards(5)) in the future. Scripts which use the `ex` family of addressing and features should use the `/usr/xpg4/bin` version of these utilities.
exec, eval, source – shell built-in functions to execute other commands

sh
  exec [argument...]
  eval [argument...]

csh
  exec command
  eval argument...
  source [-h] name

ksh
  *exec [arg...]
  *eval [arg...]

sh
  The exec command specified by the arguments is executed in place of this shell without creating a new process. Input/output arguments may appear and, if no other arguments are given, cause the shell input/output to be modified.
  
  The arguments to the eval built-in are read as input to the shell and the resulting command(s) executed.

The exec command executes command in place of the current shell, which terminates.

csh
  exec reads its arguments as input to the shell and executes the resulting command(s). This is usually used to execute commands generated as the result of command or variable substitution.

  eval reads arguments as input to the shell and executes the resulting command(s). This is usually used to execute commands generated as the result of command or variable substitution.

  source reads commands from name. source commands may be nested, but if they are nested too deeply the shell may run out of file descriptors. An error in a sourced file at any level terminates all nested source commands.

  -h Place commands from the file name on the history list without executing them.

ksh
  With the exec built-in, if arg is given, the command specified by the arguments is executed in place of this shell without creating a new process. Input/output arguments may appear and affect the current process. If no arguments are given the effect of this command is to modify file descriptors as prescribed by the input/output redirection list. In this case, any file descriptor numbers greater than 2 that are opened with this mechanism are closed when invoking another program.

  The arguments to eval are read as input to the shell and the resulting command(s) executed.

On this man page, ksh(1) commands that are preceded by one or two * (asterisks) are treated specially in the following ways:

1. Variable assignment lists preceding the command remain in effect when the command completes.
2. I/O redirections are processed after variable assignments.
3. Errors cause a script that contains them to abort.
exec(1)

4. Words, following a command preceded by ** that are in the format of a variable assignment, are expanded with the same rules as a variable assignment. This means that tilde substitution is performed after the = sign and word splitting and file name generation are not performed.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO

csh(1), ksh(1), sh(1), attributes(5)
exit, return, goto – shell built-in functions to enable the execution of the shell to advance beyond its sequence of steps

**NAME**

<table>
<thead>
<tr>
<th>sh</th>
<th>exit [n]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>return [n]</td>
</tr>
<tr>
<td>csh</td>
<td>exit [ ( expr ) ]</td>
</tr>
<tr>
<td>goto</td>
<td>label</td>
</tr>
<tr>
<td>ksh</td>
<td>*exit [n]</td>
</tr>
<tr>
<td></td>
<td>*return [n]</td>
</tr>
</tbody>
</table>

**sh**

`exit` will cause the calling shell or shell script to exit with the exit status specified by `n`. If `n` is omitted the exit status is that of the last command executed (an EOF will also cause the shell to exit.)

`return` causes a function to exit with the return value specified by `n`. If `n` is omitted, the return status is that of the last command executed.

**csh**

`exit` will cause the calling shell or shell script to exit, either with the value of the status variable or with the value specified by the expression `expr`.

The `goto` built-in uses a specified `label` as a search string amongst commands. The shell rewinds its input as much as possible and searches for a line of the form `label:` possibly preceded by space or tab characters. Execution continues after the indicated line. It is an error to jump to a label that occurs between a `while` or `for` built-in command and its corresponding `end`.

**ksh**

`exit` will cause the calling shell or shell script to exit with the exit status specified by `n`. The value will be the least significant 8 bits of the specified status. If `n` is omitted then the exit status is that of the last command executed. When `exit` occurs when executing a trap, the last command refers to the command that executed before the trap was invoked. An end-of-file will also cause the shell to exit except for a shell which has the `ignoreeof` option (See set below) turned on.

`return` causes a shell function or `.` script to return to the invoking script with the return status specified by `n`. The value will be the least significant 8 bits of the specified status. If `n` is omitted then the return status is that of the last command executed. If `return` is invoked while not in a function or a `.` script, then it is the same as an `exit`.

On this man page, `ksh(1)` commands that are preceded by one or two `*` (asterisks) are treated specially in the following ways:

1. Variable assignment lists preceding the command remain in effect when the command completes.
2. I/O redirections are processed after variable assignments.
3. Errors cause a script that contains them to abort.
4. Words, following a command preceded by ** that are in the format of a variable assignment, are expanded with the same rules as a variable assignment. This means that tilde substitution is performed after the = sign and word splitting and file name generation are not performed.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO break(1), csh(1), ksh(1), sh(1), attributes(5)
expand(1)

NAME
expand, unexpand – expand TAB characters to SPACE characters, and vice versa

SYNOPSIS
expand [-t tablist] [file...]
expand [-tabstop] [-tab1, tab2, .., tabn] [file...]
unexpand [-a] [-t tablist] [file...]

DESCRIPTION
expand copies files (or the standard input) to the standard output, with TAB characters expanded to SPACE characters. BACKSPACE characters are preserved into the output and decrement the column count for TAB calculations. expand is useful for pre-processing character files (before sorting, looking at specific columns, and so forth) that contain TAB characters.

unexpand copies files (or the standard input) to the standard output, putting TAB characters back into the data. By default, only leading SPACE and TAB characters are converted to strings of-tabs, but this can be overridden by the -a option (see the OPTIONS section below).

OPTIONS
The following options are supported for expand:

- t tablist
   Specify the tab stops. The argument tablist must consist of a single positive decimal integer or multiple positive decimal integers, separated by blank characters or commas, in ascending order. If a single number is given, tabs will be set tablist column positions apart instead of the default 8. If multiple numbers are given, the tabs will be set at those specific column positions.

   Each tab-stop position N must be an integer value greater than zero, and the list must be in strictly ascending order. This is taken to mean that, from the start of a line of output, tabbing to position N causes the next character output to be in the (N+1)th column position on that line.

   In the event of expand having to process a tab character at a position beyond the last of those specified in a multiple tab-stop list, the tab character is replaced by a single space character in the output.

- tabstop
   Specify as a single argument, sets TAB characters tabstop SPACE characters apart instead of the default 8.

- tab1, tab2, .., tabn
   Set TAB characters at the columns specified by -tab1, tab2, .., tabn

The following options are supported for unexpand:

- a
   Insert TAB characters when replacing a run of two or more SPACE characters would produce a smaller output file.
Specify the tab stops. The option-argument `tablist` must be a single argument consisting of a single positive decimal integer or multiple positive decimal integers, separated by blank characters or commas, in ascending order. If a single number is given, tabs will be set `tablist` column positions apart instead of the default 8. If multiple numbers are given, the tabs will be set at those specific column positions. Each tab-stop position `N` must be an integer value greater than zero, and the list must be in strictly ascending order. This is taken to mean that, from the start of a line of output, tabbing to position `N` will cause the next character output to be in the `(N+1)`th column position on that line. When the `-t` option is not specified, the default is the equivalent of specifying `-t 8` (except for the interaction with `-a`, described below).

No space-to-tab character conversions occur for characters at positions beyond the last of those specified in a multiple tab-stop list.

When `-t` is specified, the presence or absence of the `-a` option is ignored; conversion will not be limited to the processing of leading blank characters.

**OPERANDS**
The following operand is supported for `expand` and `unexpand`:

`file` The path name of a text file to be used as input.

**ENVIRONMENT VARIABLES**
See `environ(5)` for descriptions of the following environment variables that affect the execution of `expand` and `unexpand`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

**EXIT STATUS**
The following exit values are returned:

0 Successful completion

>0 An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

**SEE ALSO**
`tabs(1), attributes(5), environ(5)`
NAME
exportfs – translates exportfs options to share/unshare commands

SYNOPSIS
/usr/sbin/exportfs [-aiuv] [-o options] [pathname]

DESCRIPTION
exportfs translates SunOS 4.x exportfs options to the corresponding share/unshare options and invokes share/unshare with the translated options.

With no options or arguments, exportfs invokes share to print out the list of all currently shared NFS filesystems.

exportfs is the BSD/Compatibility Package command of share(1M) and unshare(1M). Use share(1M)/ unshare(1M) whenever possible.

OPTIONS
-a
  Invokes shareall(1M), or if -u is specified, invokes unshareall(1M).

-i
  Ignore options in /etc/dfs/dfstab.

-u
  Invokes unshare(1M) on pathname.

-v
  Verbose.

-o options
  Specify a comma-separated list of optional characteristics for the filesystems being exported. exportfs translates options to share-equivalent options. (see share(1M) for information about individual options).

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWnfsu</td>
</tr>
</tbody>
</table>

SEE ALSO
share(1M), shareall(1M), unshare(1M), unshareall(1M), attributes(5)
**NAME**

expr – evaluate arguments as an expression

**SYNOPSIS**

```
/usr/bin/expr argument...
/usr/xpg4/bin/expr argument...
```

**DESCRIPTION**

The `expr` utility evaluates the expression and writes the result to standard output.

The character 0 is written to indicate a zero value and nothing is written to indicate a null string.

**OPERANDS**

The `argument` operand is evaluated as an expression. Terms of the expression must be separated by blanks. Characters special to the shell must be escaped (see `sh(1)`). Strings containing blanks or other special characters should be quoted. The length of the expression is limited to LINE_MAX (2048 characters).

The operators and keywords are listed below. The list is in order of increasing precedence, with equal precedence operators grouped within `{ }` symbols. All of the operators are left-associative.

`expr \| expr`

Returns the evaluation of the first `expr` if it is neither NULL nor 0; otherwise, returns the evaluation of the second `expr` if it is not NULL; otherwise, 0.

`expr \& expr`

Returns the first `expr` if neither `expr` is NULL or 0, otherwise returns 0.

`expr[ =, \>, \>=, \,<, \,<=, \!=] expr`

Returns the result of an integer comparison if both arguments are integers, otherwise returns the result of a string comparison using the locale-specific coalition sequence. The result of each comparison will be 1 if the specified relationship is TRUE, 0 if the relationship is FALSE.

`expr \{ +, - \} expr`

Addition or subtraction of integer-valued arguments.

`expr \{ \*, /, \% \} expr`

Multiplication, division, or remainder of the integer-valued arguments.

`expr : expr`

The matching operator : (colon) compares the first argument with the second argument, which must be an internationalized basic regular expression (BRE); see `regex(5)` and `NOTES`. Normally, the `/usr/bin/expr` matching operator returns the number of bytes matched and the `/usr/xpg4/bin/expr` matching operator returns the number of characters matched (0 on failure). If the second argument contains at least one BRE sub-expression `\[ \ldots \]`, the matching operator returns the string corresponding to `\1`.

**integer**

An argument consisting only of an (optional) unary minus followed by digits.

**string**

A string argument that cannot be identified as an `integer` argument or as one of the expression operator symbols.
Compatibility Operators (IA only)

The following operators are included for compatibility with INTERACTIVE UNIX System only and are not intended to be used by non-INTERACTIVE UNIX System scripts:

- **index string character-list**
  Report the first position in which any one of the bytes in *character-list* matches a byte in *string*.

- **length string**
  Return the length (that is, the number of bytes) of *string*.

- **substr string integer-1 integer-2**
  Extract the substring of *string* starting at position *integer-1* and of length *integer-2* bytes. If *integer-1* has a value greater than the number of bytes in *string*, `expr` returns a null string. If you try to extract more bytes than there are in *string*, `expr` returns all the remaining bytes from *string*. Results are unspecified if either *integer-1* or *integer-2* is a negative value.

### EXAMPLES

**EXAMPLE 1** Adding an integer to a shell variable

Add 1 to the shell variable *a*:
```
example$ a='expr $a + 1'
```

**EXAMPLE 2** Returning a path name segment

The following example emulates `basename(1)`, returning the last segment of the path name `$a`. For `$a` equal to either `/usr/abc/file` or just `file`, the example returns `file`. (Watch out for `/` alone as an argument: `expr` takes it as the division operator; see NOTES below.)
```
example$ expr $a : '.*/(.*\)/\$a
```

**EXAMPLE 3** Using `//` characters to simplify the expression

Here is a better version of the previous example. The addition of the `//` characters eliminates any ambiguity about the division operator and simplifies the whole expression.
```
example$ expr //a : '.*/(.*)'
```

**EXAMPLE 4** Returning the number of bytes in a variable
```
example$ expr "$VAR" : '.*'
```

**EXAMPLE 5** Returning the number of characters in a variable
```
example$ expr "$VAR" : '.*'
```

### ENVIRONMENT VARIABLES

See `environ(5)` for descriptions of the following environment variables that affect the execution of `expr`: `LC_COLLATE`, `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

User Commands 367
As a side effect of expression evaluation, expr returns the following exit values:

<table>
<thead>
<tr>
<th>Exit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>If the expression is neither NULL nor 0.</td>
</tr>
<tr>
<td>1</td>
<td>If the expression is either NULL or 0.</td>
</tr>
<tr>
<td>2</td>
<td>For invalid expressions.</td>
</tr>
<tr>
<td>&gt;2</td>
<td>An error occurred.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

SEE ALSO basename(1), ed(1), sh(1), Intro(3), attributes(5), environ(5), regex(5), XPG4(5)

DIAGNOSTICS

- syntax error: Operator and operand errors.
- non-numeric argument: Arithmetic is attempted on such a string.

NOTES

After argument processing by the shell, expr cannot tell the difference between an operator and an operand except by the value. If $a$ is an =, the command:

```bash
eexpr $a = =
```

looks like:

```bash
eexpr $a = '
```

as the arguments are passed to expr (and they are all taken as the = operator). The following works:

```bash
eexpr $X$X = X= =
```

Regular Expressions

Unlike some previous versions, expr uses Internationalized Basic Regular Expressions for all system-provided locales. Internationalized Regular Expressions are explained on the regex(5) manual page.
The `expr` utility evaluates expressions as specified by its arguments. After evaluation, the result is written on the standard output. Each token of the expression is a separate argument, so terms of the expression must be separated by blanks. Characters special to the shell must be escaped. Note: 0 is returned to indicate a zero value, rather than the null string. Strings containing blanks or other special characters should be quoted. Integer-valued arguments may be preceded by a unary minus sign. Internally, integers are treated as 32-bit, two’s-complement numbers.

The operators and keywords are listed below. Characters that need to be escaped are preceded by '\'. The list is in order of increasing precedence, with equal precedence operators grouped within `{ }` symbols.

```
expr \| expr
  Returns the evaluation of the first expr if it is neither NULL nor 0; otherwise, returns
  the evaluation of the second expr if it is not NULL; otherwise, 0.
expr \& expr
  Returns the first expr if neither expr is NULL or 0, otherwise returns 0.
expr { =, \, \, \<, \<=, \!=, \=} expr
  Returns the result of an integer comparison if both arguments are integers,
  otherwise returns the result of a lexical comparison.
expr { +, - } expr
  Addition or subtraction of integer-valued arguments.
expr { \, /, \% } expr
  Multiplication, division, or remainder of the integer-valued arguments.
string : regular-expression
match string regular-expression
  The two forms of the matching operator above are synonymous. The matching
  operators : and match compare the first argument with the second argument
  which must be a regular expression. Regular expression syntax is the same as that
  of `regexp(5)`, except that all patterns are “anchored” (treated as if they begin with
  ^) and therefore ^ is not a special character, in that context. Normally, the matching
  operator returns the number of characters matched (0 on failure). Alternatively, the
  \ . . \ pattern symbols can be used to return a portion of the first argument.
substr string integer-1 integer-2
  Extracts the substring of string starting at position integer-1 and of length integer-2
  characters. If integer-1 has a value greater than the length of string, expr returns a
  null string. If you try to extract more characters than there are in string, expr
  returns all the remaining characters from string. Beware of using negative values for
  either integer-1 or integer-2 as expr tends to run forever in these cases.
```
index string character-list
   Reports the first position in string at which any one of the characters in character-list
   matches a character in string.

length string
   Returns the length (that is, the number of characters) of string.

(expr)
   Parentheses may be used for grouping.

**EXAMPLES**

**EXAMPLE 1** Adding an integer to a shell variable

Add 1 to the shell variable a.

```
a='expr $a + 1'
```

**EXAMPLE 2** Returning a path name segment

Return the last segment of a path name (that is, the filename part). Watch out for / alone as an argument: expr will take it as the division operator (see **BUGS** below).

```
# 'For $a equal to either "/usr/abc/file" or just "file"
expr $a : '..*/\ $a
```

**EXAMPLE 3** Using // characters to simplify the expression

The addition of the // characters eliminates any ambiguity about the division operator and simplifies the whole expression.

```
# A better representation of example 2.
exr //$/a : '..*/\
```

**EXAMPLE 4** Returning the value of a variable

Returns the number of characters in $VAR.

```
expr $VAR : '..*'
```

**EXIT STATUS**

expr returns the following exit codes:

<table>
<thead>
<tr>
<th>Exit Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>If the expression is neither NULL nor 0.</td>
</tr>
<tr>
<td>1</td>
<td>If the expression is NULL or 0.</td>
</tr>
<tr>
<td>2</td>
<td>For invalid expressions.</td>
</tr>
</tbody>
</table>

**ATTRIBUTES**

See attributes(5) for descriptions of the following attributes:
### SEE ALSO
sh(1), test(1), attributes(5), regexp(5)

### DIAGNOSTICS
- **syntax error** for operator/operand errors
- **non-numeric argument** if arithmetic is attempted on such a string
- **division by zero** if an attempt to divide by zero is made

### BUGS
After argument processing by the shell, `expr` cannot tell the difference between an operator and an operand except by the value. If `$a` is an `=`, the command:

```
expr $a = '
```

looks like:

```
expr = =
```

as the arguments are passed to `expr` (and they will all be taken as the `=` operator). The following works:

```
expr X$a = X=
```

Note: the `match`, `substr`, `length`, and `index` operators cannot themselves be used as ordinary strings. That is, the expression:

```bash
eexample expr index expurgatorious length
syntax error
```

generates the ‘syntax error’ message as shown instead of the value 1 as you might expect.
**NAME**
exstr – extract strings from source files

**SYNOPSIS**
exstr filename...
exstr -e filename...
exstr -r [-d] filename...

**DESCRIPTION**

The `exstr` utility is used to extract strings from C-language source files and replace them by calls to the message retrieval function (see `gettext(3C)`). This utility will extract all character strings surrounded by double quotes, not just strings used as arguments to the `printf` command or the `printf` routine. In the first form, `exstr` finds all strings in the source files and writes them on the standard output. Each string is preceded by the source file name and a colon (:).

The first step is to use `exstr -e` to extract a list of strings and save it in a file. Next, examine this list and determine which strings can be translated and subsequently retrieved by the message retrieval function. Then, modify this file by deleting lines that can’t be translated and, for lines that can be translated, by adding the message file names and the message numbers as the fourth (`msgfile`) and fifth (`msgnum`) entries on a line. The message files named must have been created by `mkmsgs(1)` and exist in `/usr/lib/locale/locale/LC_MESSAGES`. (The directory `locale` corresponds to the language in which the text strings are written; see `setlocale(3C)`). The message numbers used must correspond to the sequence numbers of strings in the message files.

Now use this modified file as input to `exstr -r` to produce a new version of the original C-language source file in which the strings have been replaced by calls to the message retrieval function `gettext()`. The `msgfile` and `msgnum` fields are used to construct the first argument to `gettext()`. The second argument to `gettext()` is printed if the message retrieval fails at run-time. This argument is the null string, unless the `-d` option is used.

This utility cannot replace strings in all instances. For example, a static initialized character string cannot be replaced by a function call. A second example is that a string could be in a form of an escape sequence which could not be translated. In order not to break existing code, the files created by invoking `exstr -e` must be examined and lines containing strings not replaceable by function calls must be deleted. In some cases the code may require modifications so that strings can be extracted and replaced by calls to the message retrieval function.

**OPTIONS**
The following options are supported:

- `-e` Extract a list of strings from the named C-language source files, with positional information. This list is produced on standard output in the following format:

  file:line:position:msgfile:msgnum:string

  `file` the name of a C-language source file
Normally you would redirect this output into a file. Then you would edit this file to add the values you want to use for msgfile and msgnum:

- msgfile the file that contains the text strings that will replace string. A file with this name must be created and installed in the appropriate place by the mkmsgs(1) utility.
- msgnum the sequence number of the string in msgfile.

The next step is to use **exstr** -r to replace strings in file.

- **-r** Replace strings in a C-language source file with function calls to the message retrieval function gettxt().
- **-d** This option is used together with the -r option. If the message retrieval fails when gettxt() is invoked at run-time, then the extracted string is printed. You would use the capability provided by **exstr** on an application program that needs to run in an international environment and have messages print in more than one language. **exstr** replaces text strings with function calls that point at strings in a message data base. The data base used depends on the run-time value of the LC_MESSAGES environment variable (see environ(5)).

### EXAMPLES

#### EXAMPLE 1

The following examples show uses of **exstr**

Assume that the file example.c contains two strings:

```c
main()
{
    printf("This is an example\n");
    printf("Hello world!\n");
}
```

The **exstr** utility, invoked with the argument example.c extracts strings from the named file and prints them on the standard output.

```
example$ exstr example.c
```

produces the following output:
EXAMPLE 1 The following examples show uses of exstr

```c
example.c:This is an example
example.c:Hello world!
```

The `exstr` utility, invoked with the `-e` option and the argument `example.c`, and redirecting output to the file `example.stringsout`

```bash
example$ exstr -e example.c > example.stringsout
```

produces the following output in the file `example.stringsout`

```c
example.c:3:8:::This is an example
example.c:4:8:::Hello world!
```

You must edit `example.stringsout` to add the values you want to use for the `msgfile` and `msgnum` fields before these strings can be replaced by calls to the retrieval function. If `UX` is the name of the message file, and the numbers 1 and 2 represent the sequence number of the strings in the file, here is what `example.stringsout` looks like after you add this information:

```c
example.c:3:8:UX:1::This is an example
example.c:4:8:UX:2::Hello world!
```

The `exstr` utility can now be invoked with the `-r` option to replace the strings in the source file by calls to the message retrieval function `gettxt()`.

```bash
example$ exstr -r example.c <example.stringsout >intlexample.c
```

produces the following output:

```c
extern char *gettxt();
main()
{
    printf(gettxt("UX:1", "");
    printf(gettxt("UX:2", "");
}
```

The following example:

```bash
example$ exstr -rd example.c <example.stringsout >intlexample.c
```

uses the extracted strings as a second argument to `gettxt()`:

```c
extern char *gettxt();
main()
{
```
EXAMPLE 1 The following examples show uses of exstr  (Continued)

```c
    printf(gettxt("UX:1", "This is an example\n"));
    printf(gettxt("UX:2", "Hello world!\n"));
```

FILES

```
/usr/lib/locale/locale/LC_MESSAGES/*
files created by mkmsgs(1)
```

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWtoo</td>
</tr>
</tbody>
</table>

SEE ALSO

gettxt(1), mkmsgs(1), printf(1), srchtxt(1), gettext(3C), printf(3C), setlocale(3C), attributes(5), environ(5)

DIAGNOSTICS

The error messages produced by exstr are intended to be self-explanatory. They indicate errors in the command line or format errors encountered within the input file.
NAME  
face – executable for the Framed Access Command Environment Interface

SYNOPSIS  
face [-i init_file] [-c command_file] [-a alias_file] [filename...]

DESCRIPTION  
The Framed Access Command Environment Interface (FACE) presents your files and file folders on the screen through a system of menus and forms if you are properly set up as a FACE user.

filename must follow the naming convention Menu.xxx for a menu, Form.xxx for a form, and Text.xxx for a text file, where xxx is any string that conforms to the UNIX system file naming conventions. The Form and Menu Language Interpreter (FMLI) descriptor lifetime will be ignored for all frames opened by argument to face. These frames have a lifetime of immortal by default. If filename is not specified on the command line, the FACE Menu will be opened along with those objects specified by the LOGINWIN environment variables. These variables are found in the user’s .environ file.

OPTIONS  
The following options are supported:

- a alias_file  Alias file
- c command_file  Command file
- i init_file  Initial file

OPERANDS  
The following operand is supported:

filename  The full pathname of the file describing the object to be opened initially.

EXIT STATUS  
The face command will return a non-zero exit value if the user is not properly set up as a FACE user.

FILES  
$HOME/pref/.environ

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWfac</td>
</tr>
</tbody>
</table>

SEE ALSO  
env(1), attributes(5)
factor(1)

NAME  
factor – obtain the prime factors of a number

SYNOPSIS  
factor [integer]

DESCRIPTION  
factor writes to standard input all prime factors for any positive integer less than or equal to $10^{14}$. The prime factors are written the proper number of times.

If factor is used without an argument, it waits for an integer to be entered. After entry of the integer, it factors it, writes its prime factors the proper number of times, and then waits for another integer. factor exits if a 0 or any non-numeric character is entered.

If factor is invoked with an argument (integer), it writes the integer, factors it and writes all the prime factors as described above, and then exits. If the argument is 0 or non-numeric, factor writes a 0 and then exits.

The maximum time to factor an integer is proportional to $\sqrt{n}$, where $n$ is the integer which is entered. factor will take this time when $n$ is prime or the square of a prime.

OPERANDS  
integer  Any positive integer less than or equal to $10^{14}$.

EXIT STATUS  
0  Successful completion.
1  An error occurred.

DIAGNOSTICS  
factor prints the error message Ouch! for input out of range or for garbage input.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
</tbody>
</table>

SEE ALSO  
attributes(5)
fastboot(1B)

NAME  fastboot, fasthalt – reboot/halt the system without checking the disks

SYNOPSIS  
  /usr/ucb/fastboot [boot-options]
  /usr/ucb/fasthalt [halt-options]

DESCRIPTION  fastboot and fasthalt are shell scripts that invoke reboot and halt with the proper arguments.

These commands are provided for compatibility only.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

SEE ALSO  fsck(1M), halt(1M), init(1M), reboot(1M), init.d(4), attributes(5)
fdformat – format floppy diskette or PCMCIA memory card

SYNOPSIS

fdformat [-dDeEfHlmMUqvx] [-b label] [-B filename] [-t dostype] [devname]

DESCRIPTION

The fdformat utility has been superseded by rmformat(1), which provides most but not all of fdformat’s functionality.

fdformat is used to format diskettes and PCMCIA memory cards. All new blank diskettes or PCMCIA memory cards must be formatted before they can be used.

fdformat formats and verifies the media and indicates whether any bad sectors were encountered. All existing data on the diskette or PCMCIA memory card, if any, is destroyed by formatting. If no device name is given, fdformat uses the diskette as a default.

By default, fdformat uses the configured capacity of the drive to format the diskette. A 3.5 inch high-density drive uses diskettes with a formatted capacity of 1.44MB. A 5.25 inch high-density drive uses diskettes with a formatted capacity of 1.2MB. In either case, a density option does not have to be specified to fdformat. However, a density option must be specified when using a diskette with a lower capacity than the drive’s default. Use the -H option to format high-density diskettes (1.44MB capacity) in an extra-high-density (ED) drive. Use the -D option, the -L option, or the -L option to format double-density (or low-density) diskettes (720KB capacity) in an HD or ED drive. To format medium-density diskettes (1.2MB capacity), use the -M option with -t nec (this is the same as using the -m option with -t nec).

Extended density uses double-sided, extended-density or extra-high-density (DS/ED) diskettes. Medium and high densities use the same media: double-sided, high-density (DS/HD) diskettes. Double (low) density uses double-sided, double-density (DS/DD) diskettes. Substituting diskettes of one density for diskettes of either a higher or lower density generally does not work. Data integrity cannot be assured whenever a diskette is formatted to a capacity not matching its density.

A PCMCIA memory card with densities from 512KB to 64MB may be formatted.

fdformat writes new identification and data fields for each sector on all tracks unless the -x option is specified. For diskettes, each sector is verified if the -v option is specified.

After formatting and verifying, fdformat writes an operating-system label on block 0. Use the -t dos option (same as the -d option) to put an MS-DOS file system on the diskette or PCMCIA memory card after the format is done. Use the -t nec option with the -M option (same as the -m option) to put an NEC-DOS file system on a diskette. Otherwise, fdformat writes a SunOS label in block 0.

OPTIONS

The following options are supported:

-`b label` Labels the media with volume label. A SunOS volume label is restricted to 8 characters. A DOS volume label is restricted to 11 upper-case characters.
fdformat(1)

- B filename
   Installs special boot loader in filename on an MS-DOS diskette.
   This option is only meaningful when the -d option (or -tdos) is
   also specified.

- D
   Formats a 720KB (3.5 inch) or 360KB (5.25 inch) double-density
   diskette (same as the -l or -L options). This is the default for
   double-density type drives. It is needed if the drive is a high- or
   extended-density type.

- e
   Ejects the diskette when done. This feature is not available on all
   systems.

- E
   Formats a 2.88MB (3.5 inch) extended-density diskette. This is the
   default for extended-density type drives.

- f
   Forces formatting, that is, this option does not ask for confirmation
   before starting format.

- H
   Formats a 1.44MB (3.5 inch) or 1.2MB (5.25 inch) high-density
   diskette. This is the default for high-density type drives; it is
   needed if the drive is the extended-density type.

- M
   Writes a 1.2MB (3.5 inch) medium-density format on a
   high-density diskette (use only with the -t nec option). This is the
   same as using -m.

   This feature is not available on all systems.

- q
   Quiet; does not print status messages.

- t dos
   Installs an MS-DOS file system and boot sector formatting. This is
   equivalent to the DOS format command or the -d option.

- t nec
   Installs an NEC-DOS file system and boot sector on the disk after
   formatting. This should be used only with the -M option. This
   feature is not available on all systems.

- U
   Performs umount on any file systems and then formats. See
   mount(1M).

- v
   Verifies each block of the diskette after the format.

- x
   Skips the format and only writes a SunOS label or an MS-DOS file
   system.

OPERANDS

The following operands are supported:

devname
   Replaces devname with rdiskette0 (systems without Volume
   Management) or floppy0 (systems with Volume Management) to
   use the first drive or rdiskette1 (systems without Volume
   Management) or floppy1 (systems with Volume Management) to
   use the second drive. If devname is omitted, the first drive, if one
   exists, is used. For PCMCIA memory cards, replace devname with
   the device name for the PCMCIA memory card which resides in
If devname is omitted, the default diskette drive, if one exists, will be used. N represents a decimal number and can be specified as follows:

cN Controller N

tN Technology type N:

```
0x1  ROM
0x2  OTPROM
0x3  EPROM
0x4  EEPROM
0x5  FLASH
0x6  SRAM
0x7  DRAM
```

dN Technology region in type N.
sN Slice N.

The following options are provided for compatibility with previous versions of fdformat. Their use is discouraged.

-\d Formats an MS-DOS floppy diskette or PCMCIA memory card (same as -t dos). This is equivalent to the MS-DOS FORMAT command.

-\l Formats a 720KB (3.5 inch) or 360KB (5.25 inch) double-density diskette (same as -D or -L). This is the default for double-density type drives; it is needed if the drive is the high- or extended-density type.

-\m Formats a 720KB (3.5 inch) or 360KB (5.25 inch) double-density diskette (same as -L or -D). This is the default for double-density type drives.

-\m Writes a 1.2 MB (3.5 inch) medium-density format on a high-density diskette (use only with the -t nec option). This is the same as using -M. This feature is not available on all systems.

FILES

```
/vol/dev/diskette0 Directory providing block device access for the media in floppy drive 0.
/vol/dev/diskette0 Directory providing character device access for the media in floppy drive 0.
/vol/dev/aliases/floppy0 Symbolic link to the character device for the media in floppy drive 0.
```
/dev/rdiskette Directory providing character device access for the media in the primary floppy drive, usually drive 0.

/vol/dev/dsk/cNTNdNs Directory providing block device access for the PCMCIA memory card. See OPERANDS for a description of N.

/vol/dev/rdsk/cNTNdNsN Directory providing character device access for the PCMCIA memory card. See OPERANDS for a description of N.

/vol/dev/aliases/pcmem Symbolic link to the character device for the PCMCIA memory card in socket S where S represents a PCMCIA socket number.

/dev/rdsk/cNTNdNs Directory providing character device access for the PCMCIA memory card. See OPERANDS for a description of N.

/dev/dsk/cNTNdNs Directory providing block device access for the PCMCIA memory card. See OPERANDS for a description of N.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO cpio(1), eject(1), rmformat(1), tar(1), volcancel(1), volcheck(1), volmissing(1), volrmmount(1), mount(1M), newfs(1M), prtvtoc(1M), vold(1M), rmmount.conf(4), vold.conf(4), attributes(5), pcfs(7FS), volfs(7FS)

IA Only fd(7D)

NOTES A diskette or PCMCIA memory card containing a ufs file system created on a SPARC based system (by using fdformat and newfs(1M)), is not identical to a diskette or PCMCIA memory card containing a ufs file system created on an IA based system. Do not interchange ufs diskettes or memory cards between these platforms. Use cpio(1) or tar(1) to transfer files on diskettes or memory cards between them. A diskette or PCMCIA memory card formatted using the -t dos option (or -d) for MS-DOS does not have the necessary system files, and is therefore not bootable. Trying to boot from it on a PC produces the following message:

Non-System disk or disk error.
Replace and strike any key when ready

BUGS Currently, bad sector mapping is not supported on floppy diskettes or PCMCIA memory cards. Therefore, a diskette or memory card is unusable if fdformat finds an error (bad sector).
The **fgrep** (fast grep) utility searches files for a character string and prints all lines that contain that string. **fgrep** is different from **grep(1)** and **egrep(1)** because it searches for a string, instead of searching for a pattern that matches an expression. It uses a fast and compact algorithm.

The characters $, *, [, ^, l, (,), and \ are interpreted literally by **fgrep**, that is, **fgrep** does not recognize full regular expressions as does **egrep**. Since these characters have special meaning to the shell, it is safest to enclose the entire **string** in single quotes `...`.

If no files are specified, **fgrep** assumes standard input. Normally, each line found is copied to the standard output. The file name is printed before each line found if there is more than one input file.

The following options are supported:

- **-b**  
  Precede each line by the block number on which it was found. This can be useful in locating block numbers by context (first block is 0).

- **-c**  
  Print only a count of the lines that contain the pattern.

- **-e pattern_list**  
  Search for a **string** in **pattern-list** (useful when the **string** begins with a `-`).

- **-f pattern-file**  
  Take the list of patterns from **pattern-file**.

- **-h**  
  Suppress printing of files when searching multiple files.

- **-i**  
  Ignore upper/lower case distinction during comparisons.

- **-l**  
  Print the names of files with matching lines once, separated by new-lines. Does not repeat the names of files when the pattern is found more than once.

- **-n**  
  Precede each line by its line number in the file (first line is 1).

- **-s**  
  Work silently, that is, display nothing except error messages. This is useful for checking the error status.

- **-v**  
  Print all lines except those that contain the pattern.

- **-x**  
  Print only lines matched entirely.

The following operands are supported:
fgrep(1)

A path name of a file to be searched for the patterns. If no file
operands are specified, the standard input will be used.

Specify a pattern to be used during the search for input.

Specify one or more patterns to be used during the search for input. This operand is treated as if it were specified as
-e pattern_list.

See largefile(5) for the description of the behavior of fgrep when encountering
files greater than or equal to 2 Gbyte (2^{31} bytes).

See environ(5) for descriptions of the following environment variables that affect the
execution of fgrep: LC_COLLATE, LC_CTYPE, LC_MESSAGES, and NLSPATH.

The following exit values are returned:

0 if any matches are found
1 if no matches are found
2 for syntax errors or inaccessible files (even if matches were found).

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWxcu4</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

SEE ALSO ed(1), egrep(1), grep(1), sed(1), sh(1), attributes(5), environ(5), largefile(5),
XPG4(5)

NOTES Ideally there should be only one grep command, but there is not a single algorithm
that spans a wide enough range of space-time tradeoffs.

Lines are limited only by the size of the available virtual memory.

The /usr/xpg4/bin/fgrep utility is identical to /usr/xpg4/bin/grep -F (see
grep(1)). Portable applications should use /usr/xpg4/bin/grep -F.
The `file` utility performs a series of tests on each file supplied by `file` and, optionally, on each file listed in `ffile` in an attempt to classify it. If the file is not a regular file, its file type is identified. The file types directory, FIFO, block special, and character special are identified as such. If the file is a regular file and the file is zero-length, it is identified as an empty file.

If file appears to be a text file, file examines the first 512 bytes and tries to determine its programming language. If file is an executable a.out, file prints the version stamp, provided it is greater than 0. If file is a symbolic link, by default the link is followed and file tests the file to which the symbolic link refers.

By default, file will try to use the localized magic file /usr/lib/locale/locale/LC_MESSAGES/magic, if it exists, to identify files that have a magic number. For example, in the Japanese locale, file will try to use /usr/lib/locale/ja/LC_MESSAGES/magic. If a localized magic file does not exist, file will utilize /etc/magic. A magic number is a numeric or string constant that indicates the file type. See magic(4) for an explanation of the format of /etc/magic.

If file does not exist, cannot be read, or its file status could not be determined, it is not considered an error that affects the exit status. The output will indicate that the file was processed, but that its type could not be determined.

The following options are supported:

- `-c` Check the magic file for format errors. For reasons of efficiency, this validation is normally not carried out.
- `-h` Do not follow symbolic links.
- `-f ffile` ffile contains a list of the files to be examined.
- `-m mfile` Use mfile as an alternate magic file, instead of /etc/magic.

The following operands are supported:

- `file` A path name of a file to be tested.

See largefile(5) for the description of the behavior of file when encountering files greater than or equal to 2 Gbyte ($2^{31}$ bytes).

**EXAMPLE 1** Binary executable files

Determine if an argument is a binary executable file:
EXAMPLE 1 Binary executable files (Continued)

```bash
file "$1" | grep -Fq executable &&
    printf "%s is executable.\n" "$1"
```

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of file: LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS

The following exit values are returned:

- 0 Successful completion.
- >0 An error occurred.

FILES

/etc/magic file’s magic number file

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

SEE ALSO

ls(1), magic(4), attributes(5), environ(5), largefile(5)

DIAGNOSTICS

If the -h option is specified and file is a symbolic link, file prints the error message:

```
symbolic link to file
```
NAME  file – determine the type of a file by examining its contents

SYNOPSIS  

/usr/ucb/file [-f file] [-cL] [-m mfile] filename...

DESCRIPTION  file performs a series of tests on each filename in an attempt to determine what it contains. If the contents of a file appear to be ASCII text, file examines the first 512 bytes and tries to guess its language.

file uses the file /etc/magic to identify files that have some sort of magic number, that is, any file containing a numeric or string constant that indicates its type.

OPTIONS  

-c  Check for format errors in the magic number file. For reasons of efficiency, this validation is not normally carried out. No file type-checking is done under -c.

-f file  Get a list of filenames to identify from file.

-L  If a file is a symbolic link, test the file the link references rather than the link itself.

-m mfile  Use mfile as the name of an alternate magic number file.

EXAMPLES  

EXAMPLE 1 Using file on all the files in a specific user's directory.

This example illustrates the use of file on all the files in a specific user's directory:

example% pwd
/usr/blort/misc
example% /usr/ucb/file *

ENVIRONMENT VARIABLES  The environment variables LC_CTYPE, LANG, and LC_default control the character classification throughout file. On entry to file, these environment variables are checked in the following order: LC_CTYPE, LANG, and LC_default. When a valid value is found, remaining environment variables for character classification are ignored. For example, a new setting for LANG does not override the current valid character classification rules of LC_CTYPE. When none of the values is valid, the shell character classification defaults to the POSIX.1 “C” locale.

FILES  /etc/magic
ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

SEE ALSO

magic(4), attributes(5)

BUGS

file often makes mistakes. In particular, it often suggests that command files are C programs.

file does not recognize Pascal or LISP.
The `filesync` utility synchronizes files between multiple computer systems, typically a server and a portable computer. `filesync` synchronizes ordinary, directory or special files. Although intended for use on nomadic systems, `filesync` is useful for backup and file replication on more permanently connected systems.

If files are synchronized between systems, the corresponding files on each of the systems are identical. Changing a file on one or both of the systems causes the files to become different (not synchronized). In order to make the files identical again, the differences between the files must be reconciled. See Reconciling and Synchronizing Files for specific details about how `filesync` reconciles and synchronizes files.

There are two forms of the `filesync` command. The first form of `filesync` is invoked without file arguments. This form of `filesync` reconciles differences between the files and systems specified in the `$HOME/.packingrules` file. `$HOME/.packingrules` is a packing rules list for `filesync` and `cachefspack`, and contains a list of files to be kept synchronized. See `packingrules(4)` and `cachefspack(1M)`.

The second form of `filesync` copies specific files from a directory on the source system to a directory on the destination system. In addition, this form of `filesync` adds the file or files specified as arguments (`filename`) to `$HOME/.packingrules`. See `-s` and `-d` for information about specifying directories on source and destination systems. See `OPERANDS` for details about specifying file (`filename`) arguments.

Multiple `filesync` commands are cumulative (that is, the specified files are added to the already existing packing rules file list). See Multiple `filesync` Commands.

`filesync` synchronizes files between computer systems by performing the following two tasks:

1. `filesync` examines the directories and files specified in the packing rules file on both systems, and determines whether or not they are identical. Any file that differs requires reconciliation.

   `filesync` also maintains a baseline summary in the `$HOME/.filesync-base` file for all of the files that are being monitored. This file lists the names, types, and sizes of all files as of the last reconciliation.

2. Based on the information contained in the baseline file and the specified options (see Resolving `filesync` Conflicts), `filesync` determines which of the various copies is the correct one, and makes the corresponding changes to the other system. Once this has been done, the two copies are, again, identical (synchronized).
If a source file has changed and the destination file has not, the changes on the source system are propagated to the destination system. If a destination file has changed and the corresponding source file has not, the changes on the destination file are propagated to the source system. If both systems have changed (and the files are not still identical) a warning message will be printed out, asking the user to resolve the conflict manually. See Resolving filesync Conflicts.

Resolving filesync Conflicts

In cases where files on both sides have changed, filesync attempts to determine which version should be chosen. If filesync cannot automatically determine which version should be selected, it prints out a warning message and leaves the two incompatible versions of the file unreconciled.

In these cases, you must either resolve the differences manually, or tell filesync how to choose which file should win. Use the -o and -f options to tell filesync how to resolve conflicts (see OPTIONS).

Alternatively, for each conflicting file, you can examine the two versions, determine which one should be kept, and manually bring the two versions into agreement (by copying, deleting, or changing the ownership or protection to be correct). You can then re-run filesync to see whether or not any other conflicts remain.

Packing Rules File

The packing rules file $HOME/.packingrules contains a list of files to be kept synchronized. The syntax of this file is described in packingrules(4).

The $HOME/.packingrules file is automatically created if users invoke filesync with filename arguments. By using filesync options, users can augment the packing rules in $HOME/.packingrules.

Many users choose to create the packing rules file manually and edit it by hand. Users can edit $HOME/.packingrules (using any editor) to permanently change the $HOME/.packingrules file, or to gain access to more powerful options that are not available from the command line (such as IGNORE commands). It is much easier to enter complex wildcard expressions by editing the $HOME/.packingrules file.

Baseline File

$HOME/.filesync-base is the filesync baseline summary file. filesync uses the information in $HOME/.filesync-base to identify the differences between files during the reconciliation and synchronization process. Users do not create or edit the baseline file. It is created automatically by filesync and records the last known state of agreement between all of the files being maintained.

Multiple filesync Commands

Over a period of time, the set of files you want to keep synchronized can change. It is common, for instance, to want to keep files pertaining to only a few active projects on your notebook. If you continue to keep files associated with every project you have ever worked on synchronized, your notebook’s disk will fill up with old files. Each filesync command will waste a lot of time updating files you no longer care about.

If you delete the files from your notebook, filesync will want to perform the corresponding deletes on the server, which would not be what you wanted. Rather, you would like a way to tell filesync to stop synchronizing some of the files. There are two ways to do this:
1. Edit $HOME/.packingrules. Delete the rules for the files that you want to delete.

2. Delete $HOME/.packingrules. Use the filesync command to specify the files that you want synchronized.

Either way works, and you can choose the one that seems easiest to you. For minor changes, it is probably easier to just edit $HOME/.packingrules. For major changes it is probably easier to start from scratch.

Once filesync is no longer synchronizing a set of files, you can delete them from your notebook without having any effect on the server.

When using filesync to keep files synchronized between nomadic machines and a server, store the packing rules and baseline files on the nomadic machines, not the server. If, when logged into your notebook, the HOME environment variable does not normally point to a directory on your notebook, you can use the FILESYNC environment variable to specify an alternate location for the packing rules and baseline files.

Each nomadic machine should carry its own packing rules and baseline file. Incorrect file synchronization can result if a server carries a baseline file and multiple nomadic machines attempt to reconcile against the server’s baseline file. In this case, a nomadic machine could be using a baseline file that does not accurately describe the state of its files. This might result in incorrect reconciliations.

To safeguard against the dangers associated with a single baseline file being shared by more than two machines, filesync adds a default rule to each new packing rules file. This default rule prevents the packing rules and baseline files from being copied.

### Nomadic Machines

### OPTIONS

The following options are supported:

- **-a**
  Force the checking of Access Control Lists (ACLs) and attempt to make them agree for all new and changed files. If it is not possible to set the ACL for a particular file, filesync stops ACL synchronization for that file.

  Some file systems do not support ACLs. It is not possible to synchronize ACLs between file systems that support ACLs and those that do not; attempting to do so will result in numerous error messages.

- **-d dest-dir**
  Specify the directory on the destination system into which filename is to be copied. Use with the -s source-dir option and the filename operand. See -s and OPERANDS.

- **-e**
  Flag all differences. It may not be possible to resolve all conflicts involving modes and ownership (unless filesync is being run with root privileges). If you cannot change the ownership or protections on a file, filesync will normally ignore conflicts in ownership and protection. If you specify the -e (everything must agree) flag, however, filesync will flag these differences.
The -f option tells filesync how to resolve conflicting changes. If a file has been changed on both systems, and an -f option has been specified, filesync will retain the changes made on the favored system and discard the changes made on the unfavored system.

Specify -f src to favor the source-system file. Specify -f dst to favor the destination-system file. Specify -f old to favor the older version of the file. Specify -f new to favor the newer version of the file.

It is possible to specify the -f and -o options in combination if they both specify the same preference (src and dst). If -f and -o conflict, the -f option is ignored. See the -o option description.

-h
Halt on error. Normally, if filesync encounters a read or write error while copying files, it notes the error and the program continues, in an attempt to reconcile other files. If the -h option is specified, filesync will immediately halt when one of these errors occurs and will not try to process any more files.

-m
Ensure that both copies of the file have the same modification time. The modification time for newly copied files is set to the time of reconciliation by default. File changes are ordered by increasing modification times so that the propagated files have the same relative modification time ordering as the original changes. Users should be warned that there is usually some time skew between any two systems, and transferring modification times from one system to another can occasionally produce strange results.

There are instances in which using filesync to update some (but not all) files in a directory will confuse the make program. If, for instance, filesync is keeping .c files synchronized, but ignoring .o files, a changed .c file may show up with a modification time prior to a .o file that was built from a prior version of the .c file.

-n
Do not really make the changes. If the -n option is specified, filesync determines what changes have been made to files, and what reconciliations are required and displays this information on the standard output. No changes are made to files, including the packing rules file.

Specifying both the -n and -o options causes filesync to analyze the prevailing system and report the changes that have been made on that system. Using -n and -o in combination is useful if your machine is disconnected (and you cannot access the server) but you want to know what changes have been made on the local machine. See the -o option description.

-o src | dst
The -o option forces a one-way reconciliation, favoring either the source system (src) or destination system (dst).
Specify `-o src` to propagate changes only from the source system to the destination system. Changes made on the destination system are ignored. `filesync` aborts if it cannot access a source or destination directory.

Specify `-o dst` to propagate changes only from the destination system to the source system. Changes made on the source system are ignored. `filesync` aborts if it cannot access a source or destination directory.

Specifying `-n` with the `-o` option causes `filesync` to analyze the prevailing system and reports on what changes have been made on that system. Using `-n` and `-o` in combination is useful if a machine is disconnected (and there is no access to the server), but you want to know what changes have been made on the local machine. See the `-n` option description.

It is possible to specify the `-o` and `-f` options in combination if they both specify the same preference (`src` or `dst`). If `-o` and `-f` options conflict, the `-f` option will be ignored. See the `-f` option description.

```
-q
```
Suppress the standard `filesync` messages that describe each reconciliation action as it is performed.

The standard `filesync` message describes each reconciliation action in the form of a UNIX shell command (for example, `mv`, `ln`, `cp`, `rm`, `chmod`, `chown`, `chgrp`, `setfacl`, and so forth).

```
-r directory
```
Limit the reconciliation to `directory`. Specify multiple directories with multiple `-r` specifications.

```
-s source-dir
```
Specify the directory on the source system from which the `filename` to be copied is located. Use with the `-d dest-dir` option and the `filename` operand. See the `-d` option description and `OPERANDS`.

```
-v
```
Display additional information about each file comparison as it is made on the standard output.

```
-y
```
Bypass safety check prompts. Nomadic machines occasionally move between domains, and many of the files on which `filesync` operates are expected to be accessed by NFS. There is a danger that someday `filesync` will be asked to reconcile local changes against the wrong file system or server. This could result in a large number of inappropriate copies and deletions. To prevent such a mishap, `filesync` performs a few safety checks prior to reconciliation. If large numbers of files are likely to be deleted, or if high level directories have changed their I-node numbers, `filesync` prompts for a confirmation before reconciliation. If you know that this is likely, and do not want to be prompted, use the `-y` (yes) option to automatically confirm these prompts.
The following operands are supported:

```
filename
```

The name of the ordinary file, directory, symbolic link, or special file in the specified source directory (`source-dir`) to be synchronized. Specify multiple files by separating each filename by spaces. Use the `filename` operand with the `-s` and `-d` options. See OPTIONS.

If `filename` is an ordinary file, that ordinary file will be replicated (with the same `filename`) in the specified destination directory (`dest-dir`).

If `filename` is a directory, that directory and all of the files and subdirectories under it will be replicated (recursively) in the specified destination directory (`dest-dir`).

If `filename` is a symbolic link, a copy of that symbolic link will be replicated in the specified destination directory (`dest-dir`).

If `filename` is a special file, a special file with the same major or minor device numbers will be replicated in the specified destination directory. (`dest-dir`). Only super-users can use `filesync` to create special files.

Files created in the destination directory (`dest-dir`) will have the same owner, group and other permissions as the files in the source directory.

If `filename` contains escaped shell wildcard characters, the wildcard characters are stored in `$HOME/.packingrules` and evaluated each time `filesync` is run.

For example, the following would make sure that the two specified files, currently in `$RHOME`, were replicated in `$HOME`:

```
filesync -s $RHOME -d $HOME a.c b.c
```

The following example would ensure that all of the `*.c` files in `$RHOME` were replicated in `$HOME`, even if those files were not created until later.

```
filesync -s $RHOME -d $HOME '*.c'
```

If any of the destination files already exist, `filesync` ensures that they are identical and issues warnings if they are not.

Once files have been copied, the distinction between the source and destination is a relatively arbitrary one (except for its use in the `-o` and `-f` switches).
variable is $HOME. The suffixes .packingrules and .filesync-base will be appended to form the names of the packing rules and baseline files.

**LC_MESSAGES**
Determines how diagnostic and informative messages are presented. In the "C" locale, the messages are presented in the default form found in the program itself (in most cases, U.S. English).

**EXIT STATUS**
Normally, if all files are already up-to-date, or if all files were successfully reconciled, `filesync` will exit with a status of 0. However, if either the -n option was specified or any errors occurred, the exit status will be the logical OR of the following:

- **0** No conflicts, all files up to date.
- **1** Some resolvable conflicts.
- **2** Some conflicts requiring manual resolution.
- **4** Some specified files did not exist.
- **8** Insufficient permission for some files.
- **16** Errors accessing packing rules or baseline file.
- **32** Invalid arguments.
- **64** Unable to access either or both of the specified src or dst directories.
- **128** Miscellaneous other failures.

**FILES**

- `$HOME/.packingrules` list of files to be kept synchronized
- `$HOME/.filesync-base` baseline summary file

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWrcmdc</td>
</tr>
</tbody>
</table>

**SEE ALSO**
cachefspack(1M), packingrules(4), attributes(5)
find(1)

NAME  
find – find files

SYNOPSIS  
/usr/bin/find path... expression
/usr/xpg4/bin/find path... expression

DESCRIPTION  
The find utility recursively descends the directory hierarchy for each path seeking files that match a Boolean expression written in the primaries given below.

find will be able to descend to arbitrary depths in a file hierarchy and will not fail due to path length limitations (unless a path operand specified by the application exceeds PATH_MAX requirements).

OPERANDS  
The following operands are supported:

path  
A path name of a starting point in the directory hierarchy.

expression  
The first argument that starts with a −, or is a ! or a (, and all subsequent arguments will be interpreted as an expression made up of the following primaries and operators. In the descriptions, wherever n is used as a primary argument, it will be interpreted as a decimal integer optionally preceded by a plus (+) or minus (−) sign, as follows:

+n   more than n
n    exactly n
-n   less than n

Expressions  
Valid expressions are:

-ath n   True if the file was accessed n days ago. The access time of directories in path is changed by find itself.

-cpio device   Always true. Writes the current file on device in cpio format (5120-byte records).

-ctime n   True if the file’s status was changed n days ago.

-depth   Always true. Causes descent of the directory hierarchy to be done so that all entries in a directory are acted on before the directory itself. This can be useful when find is used with cpio(1) to transfer files that are contained in directories without write permission.

-exec command   True if the executed command returns a zero value as exit status. The end of command must be punctuated by an escaped semicolon (;). A command argument { } is replaced by the current path name. If the last argument to -exec is { } and you specify + rather than the semicolon (;), the command will be invoked fewer times, with { } replaced by groups of pathnames.
Always true. Causes symbolic links to be followed. When following symbolic links, find keeps track of the directories visited so that it can detect infinite loops. For example, such a loop would occur if a symbolic link pointed to an ancestor. This expression should not be used with the -type l expression.

-fstype type True if the filesystem to which the file belongs is of type type.

group gname True if the file belongs to the group gname. If gname is numeric and does not appear in the /etc/group file, or in the NIS/NIS+ tables, it is taken as a group ID.

-inum n True if the file has inode number n.

-links n True if the file has n links.

-local True if the file system type is not a remote file system type as defined in the /etc/dfs/fstypes file. nfs is used as the default remote filesystem type if the /etc/dfs/fstypes file is not present. Note that -local will descend the hierarchy of non-local directories. See EXAMPLES for an example of how to search for local files without descending.

-ls Always true. Prints current path name together with its associated statistics. These include (respectively):

- inode number
- size in kilobytes (1024 bytes)
- protection mode
- number of hard links
- user
- group
- size in bytes
- modification time.

If the file is a special file, the size field will instead contain the major and minor device numbers.

If the file is a symbolic link, the pathname of the linked-to file is printed preceded by ‘→’. The format is identical to that of ls -gilds (see ls(1B)). Note: Formatting is done internally, without executing the ls program.

-mount Always true. Restricts the search to the file system containing the directory specified. Does not list mount points to other file systems.

-mtime n True if the file’s data was modified n days ago.

-name pattern True if pattern matches the current file name. Normal shell file name generation characters (see sh(1)) may be used. A backslash
( \ ) is used as an escape character within the pattern. The pattern should be escaped or quoted when `find` is invoked from the shell.

Unless the character `.` is explicitly specified in the beginning of `pattern`, a current file name beginning with `.` will not match `pattern` when using `/usr/bin/find`. `/usr/xpg4/bin/find` does not make this distinction; wildcard file name generation characters can match file names beginning with `.`.

- `ncpio device` Always true. Writes the current file on `device` in `cpio -c` format (5120 byte records).
- `newer file` True if the current file has been modified more recently than the argument `file`.
- `nogroup` True if the file belongs to a group not in the `/etc/group` file, or in the NIS/NIS+ tables.
- `nouser` True if the file belongs to a user not in the `/etc/passwd` file, or in the NIS/NIS+ tables.
- `ok command` Like `-exec`, except that the generated command line is printed with a question mark first, and is executed only if the user responds by typing y.
- `perm [-]mode` The `mode` argument is used to represent file mode bits. It will be identical in format to the symbolic mode operand, `symbolic_mode_list`, described in `chmod(1)`, and will be interpreted as follows. To start, a template will be assumed with all file mode bits cleared. An `op` symbol of:

  + Will set the appropriate mode bits in the template
  - Will clear the appropriate bits
  = Will set the appropriate mode bits, without regard to the contents of the file mode creation mask of the process

  The `op` symbol of – cannot be the first character of `mode`, to avoid ambiguity with the optional leading hyphen. Since the initial mode is all bits off, there are no symbolic modes that need to use – as the first character.

If the hyphen is omitted, the primary will evaluate as true when the file permission bits exactly match the value of the resulting template.

Otherwise, if `mode` is prefixed by a hyphen, the primary will evaluate as true if at least all the bits in the resulting template are set in the file permission bits.
### Complex Expressions

<table>
<thead>
<tr>
<th>Primary</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-perm [-]onum</code></td>
<td>True if the file permission flags exactly match the octal number <code>onum</code> (see <code>chmod(1)</code>). If <code>onum</code> is prefixed by a minus sign (<code>-</code>), only the bits that are set in <code>onum</code> are compared with the file permission flags, and the expression evaluates true if they match.</td>
</tr>
<tr>
<td><code>-print</code></td>
<td>Always true. Causes the current path name to be printed.</td>
</tr>
<tr>
<td><code>-prune</code></td>
<td>Always yields true. Does not examine any directories or files in the directory structure below the <code>pattern</code> just matched. (See EXAMPLES). If <code>-depth</code> is specified, <code>-prune</code> will have no effect.</td>
</tr>
<tr>
<td><code>-size n[c]</code></td>
<td>True if the file is <code>n</code> blocks long (512 bytes per block). If <code>n</code> is followed by a <code>c</code>, the size is in bytes.</td>
</tr>
<tr>
<td><code>-type c</code></td>
<td>True if the type of the file is <code>c</code>, where <code>c</code> is <code>b</code>, <code>c</code>, <code>d</code>, <code>D</code>, <code>f</code>, <code>l</code>, <code>p</code>, or <code>s</code> for block special file, character special file, directory, door, plain file, symbolic link, fifo (named pipe), or socket, respectively.</td>
</tr>
<tr>
<td><code>-user uname</code></td>
<td>True if the file belongs to the user <code>uname</code>. If <code>uname</code> is numeric and does not appear as a login name in the <code>/etc/passwd</code> file, or in the NIS/NIS+ tables, it is taken as a user ID.</td>
</tr>
<tr>
<td><code>-xdev</code></td>
<td>Same as the <code>-mount</code> primary.</td>
</tr>
<tr>
<td><code>-xattr</code></td>
<td>True if the file has extended attributes.</td>
</tr>
</tbody>
</table>

The primaries may be combined using the following operators (in order of decreasing precedence):

1) \(( \text{expression} )\)  
   True if the parenthesized expression is true (parentheses are special to the shell and must be escaped).  

2) \(! \text{expression}\)  
   The negation of a primary (\(!\) is the unary \textit{not} operator).  

3) \text{expression} [-a] \text{expression}  
   Concatenation of primaries (the \textit{and} operation is implied by the juxtaposition of two primaries).  

4) \text{expression} -o \text{expression}  
   Alternation of primaries (-\textit{o} is the \textit{or} operator).  

\textit{Note}: When you use `find` in conjunction with `cpio`, if you use the `-L` option with `cpio` then you must use the `-follow` expression with `find` and vice versa. Otherwise there will be undesirable results.

If no `expression` is present, `-print` will be used as the expression. Otherwise, if the given expression does not contain any of the primaries `-exec`, `-ok` or `-print`, the given expression will be effectively replaced by:

\((\text{given\_expression})\) -print
The -user, -group, and -newer primaries each will evaluate their respective arguments only once. Invocation of command specified by -exec or -ok does not affect subsequent primaries on the same file.

**USAGE**

See largefile(5) for the description of the behavior of find when encountering files greater than or equal to 2 Gbyte ($2^{31}$ bytes).

**EXAMPLES**

**EXAMPLE 1** Writing out the hierarchy directory

The following commands are equivalent:

```
example% find .
example% find . -print
```

They both write out the entire directory hierarchy from the current directory.

**EXAMPLE 2** Removing files

Remove all files in your home directory named a.out or *.o that have not been accessed for a week:

```
example% find $HOME \( -name a.out -o -name '.*.o' \) \n   -atime +7 -exec rm {} \;
```

**EXAMPLE 3** Printing all file names but skipping SCCS directories

Recursively print all file names in the current directory and below, but skipping SCCS directories:

```
example% find . -name SCCS -prune -o -print
```

**EXAMPLE 4** Printing all file names and the SCCS directory name

Recursively print all file names in the current directory and below, skipping the contents of SCCS directories, but printing out the SCCS directory name:

```
example% find . -print -name SCCS -prune
```

**EXAMPLE 5** Testing for the newer file

The following command is basically equivalent to the -nt extension to test(1):

```
example$ if [ -n "$(find file1 -prune -newer file2)" ]; then
   printf %s\n "file1 is newer than file2"
```

**EXAMPLE 6** Selecting a file using 24-hour mode

The descriptions of -atime, -ctime, and -mtime use the terminology $n$ “24-hour periods”. For example, a file accessed at 23:59 will be selected by:
EXAMPLE 6 Selecting a file using 24-hour mode (Continued)

```bash
example$ find . -atime -1 print
```

at 00:01 the next day (less than 24 hours later, not more than one day ago). The midnight boundary between days has no effect on the 24-hour calculation.

EXAMPLE 7 Printing files matching a user’s permission mode

Recursively print all file names whose permission mode exactly matches read, write, and execute access for user, and read and execute access for group and other:

```bash
example$ find . -perm u=rwx,g=rx,o=rx
```

The above could alternatively be specified as follows:

```bash
example$ find . -perm a=rwx,g-rw,o-rw
```

EXAMPLE 8 Printing files with write access for other

Recursively print all file names whose permission includes, but is not limited to, write access for other:

```bash
example$ find . -perm -o+w
```

EXAMPLE 9 Printing local files without descending non-local directories

```bash
example$ find . ! -local -prune -o -print
```

EXAMPLE 10 Printing the files in the name space possessing extended attributes

```bash
example$ find . -xattr
```

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of find: LC_COLLATE, LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS

The following exit values are returned:

- 0 All path operands were traversed successfully.
- >0 An error occurred.

FILES

- `/etc/passwd` password file
- `/etc/group` group file
- `/etc/dfs/fstypes` file that registers distributed file system packages

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:
The following options are obsolete and will not be supported in future releases:
- `cpio device` Always true. Writes the current file on device in cpio format (5120-byte records).
- `ncpio device` Always true. Writes the current file on device in cpio-c format (5120-byte records).

When using `find` to determine files modified within a range of time, use the `-mtime` argument before the `-print` argument. Otherwise, `find` will give all files.

Some files that may be under the Solaris root file system are actually mount points for virtual file systems, such as `mntfs` or `namefs`. When comparing against a ufs file system, they will not be selected if `-mount` or `-xdev` is specified in the `find` expression.
NAME
finger – display information about local and remote users

SYNOPSIS
finger [-bfhilmpqsW] [username...]

finger [-l] [username@hostname 1 [@hostname 2 .. @hostname n...]]

finger [-l] [@hostname 1 [@hostname 2 .. @hostname n...]]

DESCRIPTION
By default, the finger command displays in multi-column format the following information about each logged-in user:

- user name
- user’s full name
- terminal name (prepended with a ‘*’ (asterisk) if write-permission is denied)
- idle time
- login time
- host name, if logged in remotely

Idle time is in minutes if it is a single integer, in hours and minutes if a ‘:’ (colon) is present, or in days and hours if a ‘d’ is present.

When one or more username arguments are given, more detailed information is given for each username specified, whether they are logged in or not. username must be that of a local user, and may be a first or last name, or an account name. Information is presented in multi-line format as follows:

- the user name and the user’s full name
- the user’s home directory and login shell
- time the user logged in if currently logged in, or the time the user last logged in; and the terminal or host from which the user logged in
- last time the user received mail, and the last time the user read mail
- the first line of the $HOME/.project file, if it exists
- the contents of the $HOME/.plan file, if it exists

Note: when the comment (GECOS) field in /etc/passwd includes a comma, finger does not display the information following the comma.

If the arguments username@hostname1[@hostname2...@hostnamen] or @hostname1[@hostname2...@hostnamen] are used, the request is sent first to hostnamen and forwarded through each hostnamen-1 to hostname1. The program uses the finger user information protocol (see RFC 1288) to query that remote host for information about the named user (if username is specified), or about each logged-in user. The information displayed is server dependent.
finger(1)

As required by RFC 1288, finger passes only printable, 7-bit ASCII data. This behavior may be modified by a system administrator by using the PASS option in /etc/default/finger. Specifying PASS=low allows all characters less than decimal 32 ASCII. Specifying PASS=high allows all characters greater than decimal 126 ASCII. PASS=low,high or PASS=high,low allows both characters less than 32 and greater than 126 to pass through.

OPTIONS

The following options are supported, except that the username@hostname form supports only the -l option:

- b Suppresses printing the user’s home directory and shell in a long format printout.
- f Suppresses printing the header that is normally printed in a non-long format printout.
- h Suppresses printing of the .project file in a long format printout.
- i Forces “idle” output format, which is similar to short format except that only the login name, terminal, login time, and idle time are printed.
- l Forces long output format.
- m Matches arguments only on user name (not first or last name).
- p Suppresses printing of the .plan file in a long format printout.
- q Forces quick output format, which is similar to short format except that only the login name, terminal, and login time are printed.
- s Forces short output format.
- w Suppresses printing the full name in a short format printout.

FILES

$HOME/.plan user’s plan
$HOME/.project user’s projects
/etc/default/finger finger options file
/etc/passwd password file
/var/adm/lastlog time of last login
/var/adm/utmpx accounting

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWrcmds</td>
</tr>
</tbody>
</table>

SEE ALSO

passwd(1), who(1), whois(1), passwd(4), attributes(5)
The finger user information protocol limits the options that may be used with the remote form of this command.
**NAME**
fmlcut – cut out selected fields of each line of a file

**SYNOPSIS**

fmlcut -c list [filename...]
fmlcut -f list [-d char] [-s] [filename...]

**DESCRIPTION**
The *fmlcut* function cuts out columns from a table or fields from each line in *filename*; in database parlance, it implements the projection of a relation. *fmlcut* can be used as a filter; if *filename* is not specified or is ~, the standard input is read. *list* specifies the fields to be selected. Fields can be fixed length (character positions) or variable length (separated by a field delimiter character), depending on whether -c or -f is specified.

Note: Either the -c or the -f option must be specified.

**OPTIONS**

*list*  A comma-separated list of integer field numbers (in increasing order), with optional – to indicate ranges. For example: 1, 4, 7; 1–5, 8; –5, 10 (short for 1–5, 10); or 3– (short for third through last field).

-*c*list*  If -c is specified, *list* specifies character positions (for instance, –c1–72 would pass the first 72 characters of each line). Note: No space intervenes between -c and *list*.

-*f*list*  If -f is specified, *list* is a list of fields assumed to be separated in the file by the default delimiter character, TAB, or by *char* if the -d option is specified. For example, -f1, 7 copies the first and seventh field only. Lines with no delimiter characters are passed through intact (useful for table subheadings), unless -s is specified. Note: No space intervenes between -f and *list*. The following options can be used if you have specified -f.

-*dchar*  If -d is specified, *char* is the field delimiter. Space or other characters with special meaning to FMLI must be quoted. Note: No space intervenes between -d and *char*. The default field delimiter is TAB.

-*s*  Suppresses lines with no delimiter characters. If -s is not specified, lines with no delimiters will be passed through untouched.

**EXAMPLES**

**EXAMPLE 1** Getting login IDs and names

The following example gets the login IDs and names.

```bash
example% fmlcut -d: -f1,5 /etc/passwd
```

**EXAMPLE 2** Getting the current login name

The next example gets the current login name.

```bash
example% who am i | fmlcut -f1 -d" "
```

**ATTRIBUTES**

See attributes(5) for descriptions of the following attributes:
SEE ALSO

fmlgrep(1F), attributes(5)

DIAGNOSTICS

fmlcut returns the following exit values:

- **0** when the selected field is successfully cut out
- **2** on syntax errors

The following error messages may be displayed on the FMLI message line:

**ERROR: line too long**
A line has more than 1023 characters or fields, or there is no new-line character.

**ERROR: bad list for c / f option**
Missing `-c` or `-f` option or incorrectly specified `list`. No error occurs if a line has fewer fields than the `list` calls for.

**ERROR: no fields**
The `list` is empty.

**ERROR: no delimiter**
Missing `char` on `-d` option.

NOTES

fmlcut cannot correctly process lines longer than 1023 characters, or lines with no newline character.
The `fmlexpr` function evaluates its arguments as an expression. After evaluation, the result is written on the standard output. Terms of the expression must be separated by blanks. Characters special to FMLI must be escaped. Note that 30 is returned to indicate a zero value, rather than the null string. Strings containing blanks or other special characters should be quoted. Integer-valued arguments may be preceded by a unary minus sign. Internally, integers are treated as 32-bit, 2s complement numbers.

The operators and keywords are listed below. Characters that need to be escaped are preceded by `\`. The list is in order of increasing precedence, with equal precedence operators grouped within `{ }` symbols.

### Expressions

- **`expr | expr`**
  - Returns the first `expr` if it is neither NULL nor 0, otherwise returns the second `expr`.

- **`expr & expr`**
  - Returns the first `expr` if neither `expr` is NULL or 0, otherwise returns 0.

- **`expr { =, >, > =, <, < =, ! = } expr`**
  - Returns the result of an integer comparison if both arguments are integers, otherwise returns the result of a lexical comparison.

- **`expr { +, - } expr`**
  - Addition or subtraction of integer-valued arguments.

- **`expr { *, /, % } expr`**
  - Multiplication, division, or remainder of the integer-valued arguments.

- **`expr : expr`**
  - The matching operator `:` (colon) compares the first argument with the second argument which must be a regular expression. Regular expression syntax is the same as that of `ed(1)`, except that all patterns are "anchored" (that is, begin with `^`) and, therefore, `^` is not a special character, in that context. Normally, the matching operator returns the number of bytes matched (0 on failure). Alternatively, the `(...)` pattern symbols can be used to return a portion of the first argument.

### EXAMPLES

1. **EXAMPLE 1** Incrementing a variable
   
   Add 1 to the variable `a`:
   ```
   example% fmlexpr $a + 1 | set -l a
   ```

2. **EXAMPLE 2** Setting a variable equal to a filename
   
   For `$a` equal to either `/usr/abc/file` or just `file`:
   ```
   example% fmlexpr $a : .*/\(.*\) | $a
   ```
   
   returns the last segment of a path name (that is, `file`). Watch out for `/` alone as an argument: `fmlexpr` will take it as the division operator (see NOTES below).
**EXAMPLE 3** A better representation of Example 2

```
example% fmlexpr  //a : .*/(\.*\)
```

The addition of the // characters eliminates any ambiguity about the division operator (because it makes it impossible for the left-hand expression to be interpreted as the division operator), and simplifies the whole expression.

**EXAMPLE 4** Counting characters in a variable

Return the number of characters in $VAR:

```
example% fmlexpr $VAR : .*
```

**EXIT STATUS**

As a side effect of expression evaluation, fmlexpr returns the following exit values:

- 0 if the expression is neither NULL nor 0 (that is, TRUE)
- 1 if the expression is NULL or 0 (that is, FALSE)
- 2 for invalid expressions (that is, FALSE).

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

**SEE ALSO**

ed(1), expr(1), set(1F), sh(1), attributes(5)

**DIAGNOSTICS**

- syntax error for operator/operand errors
- non-numeric argument if arithmetic is attempted on such a string

In the case of syntax errors and non-numeric arguments, an error message will be printed at the current cursor position. Use refresh to redraw the screen.

**NOTES**

After argument processing by FMLI, fmlexpr cannot tell the difference between an operator and an operand except by the value. If $a is an =, the command:

```
example% fmlexpr $a =
```

looks like:

```
example% fmlexpr = =
```

as the arguments are passed to fmlexpr (and they will all be taken as the = operator). The following works, and returns TRUE:

```
example% fmlexpr X$a = X=
```
fmlgrep(1F)

<table>
<thead>
<tr>
<th>NAME</th>
<th>fmlgrep – search a file for a pattern</th>
</tr>
</thead>
</table>
| SYNOPSIS   | **fmlgrep** [-b] [-c] [-i] [-l] [-n] [-s] [-v] **limited_regular_expression** [filename...]
| DESCRIPTION| Fmlgrep searches filename for a pattern and prints all lines that contain that pattern. Fmlgrep uses limited regular expressions (expressions that have string values that use a subset of the possible alphanumeric and special characters) like those described on the regexp(5) manual page to match the patterns. It uses a compact non-deterministic algorithm.
Be careful when using FMLI special characters (for instance, $, ‘ , ’, ") in limited_regular_expression. It is safest to enclose the entire limited_regular_expression in single quotes ‘ . . . ’.
If filename is not specified, fmlgrep assumes standard input. Normally, each line matched is copied to standard output. The file name is printed before each line matched if there is more than one input file.
| OPTIONS    | The following options are supported:
- **-b** Precede each line by the block number on which it was found. This can be useful in locating block numbers by context (first block is 0).
- **-c** Print only a count of the lines that contain the pattern.
- **-i** Ignore upper/lower case distinction during comparisons.
- **-l** Print only the names of files with matching lines, separated by new-lines. Does not repeat the names of files when the pattern is found more than once.
- **-n** Precede each line by its line number in the file (first line is 1).
- **-s** Suppress error messages about nonexistent or unreadable files.
- **-v** Print all lines except those that contain the pattern.
| EXIT STATUS| The following exit values are returned:
0 if the pattern is found (that is, TRUE)
1 if the pattern is not found (that is, FALSE)
2 if an invalid expression was used or filename is inaccessible
| ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

410 man pages section 1: User Commands • Last Revised 28 Mar 1995
SEE ALSO
egrep(1), fgrep(1), fmlcut(1F), grep(1), attributes(5), regexp(5)

NOTES
Lines are limited to BUFSIZ characters; longer lines are truncated. BUFSIZ is defined in /usr/include/stdio.h.

If there is a line with embedded nulls, fmlgrep will only match up to the first null; if it matches, it will print the entire line.
fmli(1)

NAME
fmli – invoke FMLI

SYNOPSIS
fmli [-a alias_file] [-c command_file] [-i initialization_file] filename...

DESCRIPTION
The fmli command invokes the Form and Menu Language Interpreter and opens the frame(s) specified by the filename argument. The filename argument is the pathname of the initial frame definition file(s), and must follow the naming convention Menu.xxx, Form.xxx, or Text.xxx for a menu, form or text frame respectively, where xxx is any string that conforms to UNIX system file naming conventions. The FMLI descriptor lifetime will be ignored for all frames opened by argument to fmli. These frames have a lifetime of immortal by default.

OPTIONS
The following options are supported:

-a alias_file
If -a is specified, alias_file is the name of a file which contains lines of the form alias=pathname. Thereafter, $alias can be used in definition files to simplify references to objects or devices with lengthy pathnames, or to define a search path (similar to $PATH in the UNIX system shell).

-c command_file
If -c is specified, command_file is the name of a file in which default FMLI commands can be disabled, and new application-specific commands can be defined. The contents of command_file are reflected in the FMLI Command Menu.

-i initialization_file
If -i is specified, initialization_file is the name of a file in which the following characteristics of the application as a whole can be specified:

- A transient introductory frame displaying product information
- A banner, its position, and other elements of the banner line
- Color attributes for all elements of the screen
- Screen Labeled Keys (SLKs) and their layout on the screen.

EXAMPLES
EXAMPLE 1 Examples of the fmli command.

To invoke fmli:

example$ fmli Menu.start

where Menu.start is an example of filename named according to the file name conventions for menu definition files explained above.

To invoke fmli and name an initialization file:
EXAMPLE 1 Examples of the fmli command. (Continued)

example$ fmli -i init.myapp Menu.start

where init.myapp is an example of initialization file.

Variables

LOADPFK Leaving this environment variable unset tells FMLI, for certain terminals like the AT&T 5620 and 630, to download its equivalent character sequences for using function keys into the terminal’s programmable function keys, wiping out any settings the user may already have set in the function keys. Setting LOADPFK=NO in the environment will prevent this downloading.

COLUMNS Can be used to override the width of the logical screen defined for the terminal set in TERM. For terminals with a 132-column mode, for example, invoking FMLI with the line

COLUMNS=132 fmli frame-file

will allow this wider screen width to be used.

LINES Can be used to override the length of the logical screen defined for the terminal set in TERM.

FILES /usr/bin/fmli

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO vsig(1F), attributes(5)

DIAGNOSTICS If filename is not supplied to the fmli command, fmli returns the message:

Initial object must be specified.

If filename does not exist or is not readable, fmli returns an error message and exits. The example command line above returns the following message and exits:

Can’t open object "Menu.start"

If filename exists, but does not start with one of the three correct object names (Menu., Form., or Text.) or if it is named correctly but does not contain the proper data, fmli starts to build the screen by putting out the screen labels for function keys, after which it flashes the message:
I do not recognize that kind of object

and then exits.
**fmt(1)**

**NAME**
fmt – simple text formatters

**SYNOPSIS**
fmt [-cs] [-w width | -width] [inputfile...]

**DESCRIPTION**
fmt is a simple text formatter that fills and joins lines to produce output lines of (up to) the number of characters specified in the 
-w width option. The default width is 72.
fmt concatenates the inputfiles listed as arguments. If none are given, fmt formats text from the standard input.

Blank lines are preserved in the output, as is the spacing between words. fmt does not fill nor split lines beginning with a ’.’ (dot), for compatibility with nroff(1). Nor does it fill or split a set of contiguous non-blank lines which is determined to be a mail header, the first line of which must begin with ”From”.

Indentation is preserved in the output, and input lines with differing indentation are not joined (unless -c is used).

fmt can also be used as an in-line text filter for vi(1). The vi command:

`%!fmt`

reformats the text between the cursor location and the end of the paragraph.

**OPTIONS**
- `-c` Crown margin mode. Preserve the indentation of the first two lines within a paragraph, and align the left margin of each subsequent line with that of the second line. This is useful for tagged paragraphs.

- `-s` Split lines only. Do not join short lines to form longer ones. This prevents sample lines of code, and other such formatted text, from being unduly combined.

- `-w width | -width` Fill output lines to up to width columns.

**OPERANDS**
inputfile Input file.

**ENVIRONMENT VARIABLES**
See environ(5) for a description of the LC_CTYPE environment variable that affects the execution of fmt.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

**SEE ALSO**
nroff(1), vi(1), attributes(5), environ(5)

**NOTES**
The -width option is acceptable for BSD compatibility, but it may go away in future releases.
fmtmsg(1)

NAME
fmtmsg – display a message on stderr or system console

SYNOPSIS

DESCRIPTION
Based on a message’s classification component, fmtmsg either writes a formatted message to stderr or writes a formatted message to the console.

A formatted message consists of up to five standard components (see environment variable MSGVERB in the ENVIRONMENT VARIABLES section of this page). The classification and subclass components are not displayed as part of the standard message, but rather define the source of the message and direct the display of the formatted message.

OPTIONS
The following options are supported:

- **-c class**
  Describes the source of the message. Valid keywords are:

  - **hard**
    The source of the condition is hardware.

  - **soft**
    The source of the condition is software.

  - **firm**
    The source of the condition is firmware.

- **-u subclass**
  A list of keywords (separated by commas) that further defines the message and directs the display of the message. Valid keywords are:

  - **appl**
    The condition originated in an application. This keyword should not be used in combination with either *util* or *opsys*.

  - **util**
    The condition originated in a utility. This keyword should not be used in combination with either *appl* or *opsys*.

  - **opsys**
    The message originated in the kernel. This keyword should not be used in combination with either *appl* or *util*.

  - **reco**
    The application will recover from the condition. This keyword should not be used in combination with *nreco*.

  - **nreco**
    The application will not recover from the condition. This keyword should not be used in combination with *reco*.

  - **print**
    Print the message to the standard error stream stderr.

  - **console**
    Write the message to the system console. *print*, *console*, or both may be used.
### fmtmsg(1)

- `-l label` Identifies the source of the message.
- `-s severity` Indicates the seriousness of the error. The keywords and definitions of the standard levels of `severity` are:
  - `halt` The application has encountered a severe fault and is halting.
  - `error` The application has detected a fault.
  - `warn` The application has detected a condition that is out of the ordinary and might be a problem.
  - `info` The application is providing information about a condition that is not in error.
- `-t tag` The string containing an identifier for the message.
- `-a action` A text string describing the first step in the error recovery process. This string must be written so that the entire `action` argument is interpreted as a single argument. `fmtmsg` precedes each action string with the `TO FIX:` prefix.
- `text` A text string describing the condition. Must be written so that the entire `text` argument is interpreted as a single argument.

#### EXAMPLE 1 Standard message format

The following example of `fmtmsg` produces a complete message in the standard message format and displays it to the standard error stream.

```bash
example$ fmtmsg -c soft -u recov,print,appl -l UX:cat \
   -s error -t UX:cat:001 -a "refer to manual" "invalid syntax"
```

produces:

```
UX:cat: ERROR: invalid syntax
TO FIX: refer to manual  UX:cat:138
```

#### EXAMPLE 2 Using MSGVERB

When the environment variable `MSGVERB` is set as follows:

```
MSGVERB=severity:text:action
```

and Example 1 is used, `fmtmsg` produces:

```
ERROR: invalid syntax
TO FIX: refer to manual
```
EXAMPLE 2 Using MSGVERB (Continued)

EXAMPLE 3 Using SEV_LEVEL

When the environment variable SEV_LEVEL is set as follows:

```
SEV_LEVEL=note,5,NOTE
```

the following fmtmsg command:

```
example% fmtmsg -c soft -u print -l UX:cat -s note -a "refer to manual" "invalid syntax"
```

produces:

```
NOTE: invalid syntax
TO FIX: refer to manual
```

and displays the message on stderr.

The environment variables MSGVERB and SEV_LEVEL control the behavior of fmtmsg. MSGVERB is set by the administrator in the /etc/profile for the system. Users can override the value of MSGVERB set by the system by resetting MSGVERB in their own .profile files or by changing the value in their current shell session. SEV_LEVEL can be used in shell scripts.

MSGVERB tells fmtmsg which message components to select when writing messages to stderr. The value of MSGVERB is a colon-separated list of optional keywords. MSGVERB can be set as follows:

```
MSGVERB=[keyword [keyword [...]]]
export MSGVERB
```

Valid keywords are: label, severity, text, action, and tag. If MSGVERB contains a keyword for a component and the component’s value is not the component’s null value, fmtmsg includes that component in the message when writing the message to stderr. If MSGVERB does not include a keyword for a message component, that component is not included in the display of the message. The keywords may appear in any order. If MSGVERB is not defined, if its value is the null string, if its value is not of the correct format, or if it contains keywords other than the valid ones listed above, fmtmsg selects all components.

MSGVERB affects only which message components are selected for display. All message components are included in console messages.

SEV_LEVEL defines severity levels and associates print strings with them for use by fmtmsg. The standard severity levels shown below cannot be modified. Additional severity levels can be defined, redefined, and removed.

```
0 (no severity is used)
```
SEV_LEVEL is set as follows:

description is a comma-separated list containing three fields:

```
SEV_LEVEL= [description : description [:...]]
```

```
export SEV_LEVEL

description=severity_keyword, level, printstring
```

severity_keyword is a character string used as the keyword with the \texttt{-s} severity option to 
\texttt{fmtmsg}.

level is a character string that evaluates to a positive integer (other than 0, 1, 2, 3, or 4, 
which are reserved for the standard severity levels). If the keyword \texttt{severity_keyword} is 
used, level is the severity value passed on to \texttt{fmtmsg}(3C).

printstring is the character string used by \texttt{fmtmsg} in the standard message format 
whenever the severity value level is used.

If \texttt{SEV\_LEVEL} is not defined, or if its value is null, no severity levels other than the 
defaults are available. If a description in the colon separated list is not a comma 
separated list containing three fields, or if the second field of a comma separated list 
does not evaluate to a positive integer, that description in the colon separated list is 
ignored.

**EXIT STATUS** The following exit values are returned:

0 All the requested functions were executed successfully.

1 The command contains a syntax error, an invalid option, or an invalid 
argument to an option.

2 The function executed with partial success, however the message was not 
displayed on \texttt{stderr}.

4 The function executed with partial success; however, the message was not 
displayed on the system console.

32 No requested functions were executed successfully.

**ATTRIBUTES** See \texttt{attributes(5)} for descriptions of the following attributes:

fmtmsg(1)
fmtmsg(1)

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

**SEE ALSO**
addseverity(3C), fmtmsg(3C), attributes(5)
### NAME
fnattr – update and examine attributes associated with an FNS named object

### SYNOPSIS
```
fnattr [-AL] composite_name [ [-O | -U] identifier...]
fnattr [-L] composite_name [ { -a [-s] [-O | -U] identifier [value...] } | { -d
   [ [-O | -U] identifier [value...] ] | { -m [-O | -U] identifier old_value
   new_value }... ]
```

### DESCRIPTION
The `fnattr` command is for updating and examining attributes associated with an FNS named object. There are four uses for this command: add an attribute or value, delete an attribute or value, modify an attribute’s value, and list the contents of an attribute.

### OPTIONS
The options for adding, modifying, and deleting attributes and their values can be combined in the same command line. The modifications will be executed in the order that they are specified.

Any unsuccessful modification will abort all subsequent modifications specified in the command line; any modifications already carried out will remain. The unsuccessful modifications are displayed as output of `fnattr`.

- `-a` Add an attribute or add a value to an attribute associated with object named by `composite_name`. `identifier` is the identifier of the attribute to manipulate; its format is `FN_ID_STRING` unless the `-O` or `-U` option is given. `value...` represents the attribute values to add. The attribute syntax used for storing `value` is `fn_attr_syntax_ascii`.

- `-A` Consult the authoritative source to get attribute information.

- `-d` Delete attributes associated with object named by `composite_name`. If `identifier` is not specified, all attributes associated with the named object are deleted. If `identifier` is specified without accompanying values (`value...`), the entire attribute identified by `identifier` is removed. If individual attribute values (`value...`) are specified, then only these are removed from the attribute. Removal of the last value of an attribute entails removal of the attribute as well. The format of `identifier` is `FN_ID_STRING` unless the `-O` or `-U` option is given.

- `-L` If the composite name is bound to an XFN link, manipulate the attributes associated with the object pointed to by the link. If `-L` is not used, the attributes associated with the XFN link are manipulated.

- `-m` Modify the values of the attribute identified by `identifier associated with the object named by `composite_name`. `old_value` is replaced by `new_value` in the specified attribute. Other attributes and values associated with `composite_name` are not affected. The format of `identifier` is `FN_ID_STRING` unless the `-O` or `-U` option is given.

- `-O` The format of `identifier` is `FN_ID_ISO_OID_STRING`, an ASN.1 dot-separated integer list string.
Add in supersede mode. If an attribute with the same identifier as `identifier` already exists, remove all its values, and replace with `value`. If this option is omitted, the resulting values for the specified attribute is a union of the existing values and `value`.

The format of `identifier` is `FN_ID_DCE_UUID`, a DCE UUID in string form.

**OPERANDS**
The following operand is supported:

- `composite_name` An FNS named object.

**Adding**
The `-a` option is used for adding attributes and values. This following command replaces the value of the `shoesize` attribute of `user/jane` with the value 7.5:

```
eg% fnattr user/jane -as shoesize 7.5
```

The following command adds the value `Chameleo` to the `project` attribute of `user/jane`:

```
eg% fnattr user/jsmith -a project Chameleo
```

**Deleting**
The `-d` option is used for deleting attributes and values. The following command deletes all the attributes associated with `user/jane`:

```
eg% fnattr user/jane -d
```

The following command deletes the attribute `shoesize` associated with `user/jane`:

```
eg% fnattr user/jane -d shoesize
```

The following command deletes the attribute value `old_project` from the `projects` attribute associated with `user/jane`:

```
eg% fnattr user/jane -d projects old_project
```

**Modifying**
The `-m` option is for modifying an attribute value. The following command replaces the value `Chameleo` by `Dungeon` in the `projects` attribute associated with `user/jsmith`:

```
eg% fnattr user/jsmith -m projects Chameleo Dungeon
```

The following command is an example of unsuccessful modification attempts. The user executing this command does not have permission to update `user/jane`’s attributes but is allowed to add new attributes. Executing the command will add the attribute `hatsize` but will not delete `shoesize` or modify `dresssize` because `-d` will fail and cause the command to stop:
Listing

No options are required to list attributes and their values. The following command lists all the attributes associated with user/jane:

```
eg% fnattr user/jane
```

The following command lists the values of the project attribute of user/jane:

```
eg% fnattr user/jane project
```

The following command lists the values of the project and shoesize attributes of user/jane:

```
eg% fnattr user/jane project shoesize
```

EXIT STATUS

0 Operation was successful.
1 Operation failed.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWfns</td>
</tr>
</tbody>
</table>

SEE ALSO

fnlookup(1), attributes(5), fns(5)

NOTES

Built-in attributes, such as onc_unix_passwd for users, cannot be updated using the fnattr command. Their contents are affected by updates to the underlying naming service, such as NIS+ or NIS.
fnbind(1)

NAME  
fnbind – Bind a reference to an FNS name

SYNOPSIS  
fnbind  [-s] [-v] [-L]  name  new_name

fnbind  -r [-s] [-v]  new_name  [-O | -U]  ref_type  { [-O | -U]  addr_type  [-c | -x]  addr_contents } |

DESCRIPTION  
The fnbind utility binds the reference named by name to the name new_name. The second synopsis of fnbind (uses the -r option) allows the binding of new_name to the reference constructed using arguments supplied in the command line.

OPTIONS  
The following options are supported:

- s  Bind to new_name even if it is already bound. If this option is omitted, fnbind fails if new_name is already bound.
- v  Display the reference being bound to new_name.
- L  Create an XFN link using name and bind it to new_name.
- r  Create a reference using ref_type as the reference’s type, and one or more pairs of addr_type and addr_contents as the reference’s list of addresses, and bind this reference to new_name. Unless the -O or -U options are used, FN_ID_STRING is used as the identifier format for ref_type and addr_type. Unless the -c or -x options are used, addr_contents is stored as an XDR-encoded string.
- c  Store addr_contents in the given form; do not use XDR-encoding.
- x  addr_contents specifies a hexadecimal string. Convert it to its hexadecimal representation and store it; do not use XDR-encoding.
- O  The identifier format is FN_ID_ISO_OID_STRING, an ASN.1 dot-separated integer list string.
- U  The identifier format is FN_ID_DCE_UUID, a DCE UUID in string form.

EXAMPLES  
EXAMPLE 1 Binding a service to a printer
The command

example% fnbind -s thisorgunit/service/printer thisorgunit/service/pr

binds the name thisorgunit/service/pr to the reference named by thisorgunit/service/printer. Any reference bound to thisorgunit/service/printer is overwritten.

EXAMPLE 2 Binding to an XFN link
The command

example% fnbind -L thisorgunit/service/printer thisorgunit/service/pr
EXAMPLE 2 Binding to an XFN link

(Continued)

binds the name thisorgunit/service/pr to the XFN link constructed using the name thisorgunit/service/printer.

EXAMPLE 3 Binding to an address type

The command

```bash
example$ fnbind -r thisorgunit/service/calendar SUNW_cal SUNW_cal_deskset_onc staff@exodus
```

binds the name thisorgunit/service/calendar to the reference with reference type SUNW_cal and address type SUNW_cal_deskset_onc, and address contents of staff@exodus.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWfns</td>
</tr>
</tbody>
</table>

SEE ALSO fnlookup(1), fnrename(1), fnunbind(1), FN_identifier_t(3XFN), xdr(3NSL), attributes(5), fns(5), xfn_links(3XFN)
fnlist(1)

NAME  fnlist – display the names and references bound in an FNS context

SYNOPSIS  fnlist [-A lv] [composite_name]

DESCRIPTION  fnlist displays the names and references bound in the context of composite_name. If composite_name is not provided, the default initial context is displayed.

OPTIONS  The following options are supported:

- A Consult the authoritative source for information.
- l Display the references as well as the names bound in the context of composite_name. Without this option, only the names are displayed.
- v Display the references in detail. For onc_fn_* references, this option is useful to derive the name of the NIS+ table that stores the reference for every name bound in the context of composite_name.

OPERANDS  The following operand is supported:

composite_name  An FNS named object. Composite names, like UNIX file names, depend on the subcontexts created. Examples of commands with valid composite_name operands are:

eg% fnlist thisorgunit
eg% fnlist thisorgunit/service
eg% fnlist thisorgunit/service/printer

When FNS is deployed, the composite name is specific to the deployed site.

EXAMPLES  EXAMPLE 1 Examples of the fnlist command.

In the following example, the command with no operand provides the listing with reference and address types for the initial context:

eg% fnlist -l

In the following examples, where a user context is given (that is, composite_name = user/), FNS must first be deployed via fncreate(1M), using one of the naming services NIS, NIS+, or files. If FNS is not deployed, there are no user contexts and the commands will fail with the "Name not found" error message.

The following command shows the names bound in the context of user/:

eg% fnlist user/

The following command displays the names and references bound in the context of user/:

eg% fnlist -l user/
EXAMPLE 1 Examples of the fnlist command. (Continued)

EXIT STATUS
0 Operation was successful.
1 Operation failed.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWfns</td>
</tr>
</tbody>
</table>

SEE ALSO fnbind(1), fnlookup(1), fnunbind(1), fncreate(1M), fndestroy(1M), attributes(5), fns(5), fns_references(5)
fnlookup(1)

NAME  fnlookup – display the reference bound to an FNS name

SYNOPSIS  fnlookup [-ALv] composite_name

DESCRIPTION  fnlookup displays the binding of composite_name.

OPTIONS  

  -A  Consult the authoritative source for information.
  -L  If the composite name is bound to an XFN link, display the reference that
       the link is bound to. Without the -L option, fnlookup displays the XFN
       link.
  -v  Display the binding in detail. For onc_fn_* references, this option is
       useful to derive the name of the NIS+ table that stores the reference for
       composite_name and a string representation of the reference, if applicable.

OPERANDS  The following operand is supported:

  composite_name  An FNS named object.

EXAMPLES  EXAMPLE 1 Examples of the fnlookup command.

In the following example, the command

  eg% fnlookup user/jsmith/service/calendar

shows the reference to which the name user/jsmith/service/calendar, that
refers to the calendar of user jsmith, is bound.

In the next example, the command

  eg% fnlookup user/jsmith/service

shows the reference to which the name user/jsmith/service, that refers to the
service context of user jsmith, is bound. If this is bound to an XFN link, then

  eg% fnlookup -L user/jsmith/service

displays the reference to which this link is bound.

EXIT STATUS  

  0  Operation was successful.
  1  Operation failed.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWfns</td>
</tr>
</tbody>
</table>

SEE ALSO  fnbind(1), fnlist(1), fnunbind(1), fncreate(1M), fndestroy(1M),
           xfn_links(3XFN), attributes(5), fns(5), fns_references(5)
fnrename(1)

NAME
fnrename – rename the binding of an FNS name

SYNOPSIS
fnrename [-s] [-v] context_name old_atomic_name new_atomic_name

DESCRIPTION
fnrename renames the binding of old_atomic_name to new_atomic_name in the context of context_name. Both old_atomic_name and new_atomic_name must be atomic names, to be resolved in the context named by context_name.

OPTIONS
- s Overwrite any reference already bound to new_atomic_name. If this option is omitted, fnrename fails if new_atomic_name is already bound.
- v Display the binding being renamed.

EXAMPLES
EXAMPLE 1 An example of the fnrename command.
For example, the command

eg% fnrename user/jsmith/service/ clendar calendar

binds calendar to the reference bound to clendar in the context named by user/jsmith/service/ and unbinds clendar.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWfns</td>
</tr>
</tbody>
</table>

SEE ALSO
fnbind(1), fnlist(1), fnunbind(1), fncreate(1M), fndestroy(1M), xfn_links(3XFN), attributes(5), fns(5), fns_references(5)
fnsearch(1)

NAME
fnsearch – search for FNS objects with specified attributes

SYNOPSIS
fnsearch [-Allv] [-n max] [-s scope] composite_name [-a ident]... [-O | -U] filter_expr [filter_arg]...

DESCRIPTION
The fnsearch command operation displays the names and, optionally, the attributes and references of objects bound at or below composite_name whose attributes satisfy a given filter expression. The filter expression is given in terms of logical expressions involving the identifiers and values of the attributes and references of objects examined during the search.

For general information about FNS, see fna(5).

OPTIONS
The following options are supported:

- **a ident**
  Display the given attribute of each object that satisfies the filter expression. If the -a option is not used, all attributes are displayed. An empty ident (" " from the shell) indicates that no attributes are to be displayed. Multiple -a options may be given. The syntax of ident is described fully under Displaying Selected Attributes below.

- **A**
  Consult the authoritative source(s) for information.

- **l**
  Display the reference of each object that satisfies the filter expression.

- **L**
  Follow XFN links during the search.

- **n max**
  Restrict the maximum number of objects displayed to the given number (a positive integer). There is no limit by default.

- **s scope**
  Set the scope of the search. scope is one of:

  - **object**
    Only the object composite_name is searched.

  - **context**
    Objects bound directly to composite_name are searched.

  - **subtree**
    Objects bound to composite_name or any of its subcontexts are searched.

  - **constrained_subtree**
    Like subtree, but the search may be restricted to a set of subcontexts defined in a context-implementation-defined manner.
scope may be abbreviated to any unambiguous prefix, such as \( \circ \) or \( \text{cont} \). If this option is not given, the default behavior is \(-s\) context.

\(-v\) Display in detail the reference of each object that satisfies the filter expression. This option takes precedence over \(-l\).

**OPERANDS**

The following operand is supported:

- \textit{composite\_name} An FNS named object.

**Simple Filter Expressions**

The simplest form of filter expression is one that tests for the existence of an attribute. This expression is formed simply by giving the attribute’s name. To search for objects having an attribute named \texttt{for\_sale}, for example:

\[ \% \text{fnsearch composite\_name for\_sale} \]

Another simple filter expression is one that tests the value of a particular attribute. To find objects whose ages are less than 17:

\[ \% \text{fnsearch composite\_name "age < 17"} \]

String values are indicated by enclosing the string in single quotes. To find all \texttt{red} objects:

\[ \% \text{fnsearch composite\_name "color == 'red'"} \]

Note that the double quotes (" ") in this example are not part of the filter expression. Instead, they prevent the shell from interpreting the white-space and single quotes that are part of the expression.

**Logical Operators**

Simple filter expressions may be composed using the logical operators \texttt{and}, \texttt{or}, and \texttt{not}. For example:

\[ \% \text{fnsearch composite\_name "age >= 35 and us\_citizen"} \]

Parentheses may be used to group expressions:

\[ \% \text{fnsearch composite\_name "not (make == 'olds' and year == 1973)"} \]

The precedence of operators is, in order of increasing precedence:

\texttt{or} and \texttt{not} relational operators (see Relational Operators below)

The logical operators \texttt{and} and \texttt{or} are left-associative.

**Relational Operators**

The following are the relational operators that may be used to compare an attribute to a supplied value:

\texttt{==} True if at least one value of the attribute is equal to the supplied value.

\texttt{!=} True if none of the attribute’s values are equal to the supplied value.
True if at least one value of the attribute is less than the supplied value.

<= True if at least one value of the attribute is less than or equal to the supplied value.

> True if at least one value of the attribute is greater than the supplied value.

>= True if at least one value of the attribute is greater than or equal to the supplied value.

~= True if at least one value of the attribute matches the supplied value according to some context-specific approximate matching criterion. This criterion must subsume strict equality.

Comparisons and ordering are specific to the syntax or rules of the attribute being tested.

By default, the fnsearch command displays the names and all of the attributes of each object matching the search criteria. The list of attributes displayed may be restricted by using the -a command line option. In the following example, only the color and shape attributes of small objects are displayed:

```
% fnsearch composite_name -a color -a shape "size == 'small'"
```

The format of an attribute identifier is taken to be FN_ID_STRING (an ASCII string) by default. To name an attribute identifier that is an OSI OID or a DCE UUID, the attribute name is prefixed by -O or -U, respectively:

- O The identifier format is FN_ID_ISO_OID_STRING, an ASN.1 dot-separated integer list string.

- U The identifier format is FN_ID_DCE_UUID, a DCE UUID in string form.

For example:

```
% fnsearch composite_name -a -O 2.5.4.0 "shoe_size < 9"
```

and

```
% fnsearch composite_name -a -U 0006a446-5e97-105f-9828-8190285baa77 "bowling_avg > 200"
```

Some parts of a filter expression may be replaced by a substitution token: a percent sign (%) followed by a single character. The value of this portion of the expression is then given in a filter argument that follows the filter expression, in much the same way as is done in printf(1). The available substitution tokens are:

- %a attribute
- %s string
- %i identifier
- %v attribute value (the only syntax currently supported is fn_attr_syntax_ascii)
For example, the command:
```
$ fnsearch composite_name *color == 'red'
```
could equivalently be written:
```
$ fnsearch composite_name "%a == 'red'" color
```
or:
```
$ fnsearch composite_name "%a == %s" color red
```

The use of substitution tokens is helpful when writing shell scripts in which the values of the filter arguments are generated at run-time.

By default, the format of the identifier of an attribute such as the `color` attribute above is taken to be `FN_ID_STRING` (an ASCII string). Substitution tokens enable the use of OSI OIDs and DCE UUIDs instead. The filter argument is prefixed by `-O` or `-U`, with the same meaning as in the `-a` command line option described above:

- `-O`: The identifier format is `FN_ID_ISO_OID_STRING`, an ASN.1 dot-separated integer list string.
- `-U`: The identifier format is `FN_ID_DCE_UUID`, a DCE UUID in string form.

For example:
```
$ fnsearch composite_name "%a = 2.5.4.0
```

and
```
$ fnsearch composite_name "%a == 'red'" \ 
   -U 0006a446-5e97-105f-9828-8190285baa77
```

Wildcarded Strings

A wildcarded string consists of a sequence of alternating wildcard specifiers and strings. The wildcard specifiers is denoted by the asterisk (`*`) and means zero or more occurrences of any character.

Wildcarded strings are used to specify substring matches. The following are some examples of wildcarded strings and their meanings.

```
*                any string
'tom'            the string "tom"
'hav'            any string starting with "hav"
'ing'            any string ending with "ing"
'a'**'b'         any string starting with "a" and ending with "b"
'jo'**'ph'**'ne''er' any string starting with "jo" and containing the substring "ph", and which contains the substring "ne" in the portion of the string following "ph", and which ends with "er"
```
Extended Operations

Extended operators are predicates (functions that return TRUE or FALSE) that may be freely mixed with other operators in a filter expression.

An extended operation is specified by giving the operation name as a quoted string, followed by an argument in parentheses. The following three extended operations are currently defined:

- `'name' (WildcardedString)` \( \text{TRUE if the name of the object matches the supplied wildcarded string.} \)
- `'reftype' (Identifier)` \( \text{TRUE if the reference type of the object is equal to the supplied identifier.} \)
- `'addrtype' (Identifier)` \( \text{TRUE if any of the address types in the reference of the object are equal to the supplied identifier.} \)

The following example shows a search for objects whose names start with `bill` and having IQ attributes over 80:

```
\$ fnsearch composite_name "name('bill*') and IQ > 80"
```

The complete grammar of filter expressions is given below. It is based on the grammar defined by the XFN specification (see `FN_search_filter_t(3XFN)`).

String literals in this grammar are enclosed in double quotes; the quotes are not themselves part of the expression. Braces are used for grouping; brackets indicate optional elements. An unquoted asterisk (*) signifies zero or more occurrences of the preceding element; a plus sign (+) signifies one or more occurrences.

```
FilterExpr : : = [Expr]

Expr : : =

    Expr "or" Expr | Expr "and" Expr | "not" Expr | "(" Expr ")" | Attribute [RelOp Value] | Ext

RelOp : : =

    ":=" | ":" | ":<" | ":<=" | ":>" | ":>=" | ":-="

Attribute : : =

    Char* | "%a"

Value : : =

    Integer | WildcardedString | "%v"
```
**fnsearch(1)**

```plaintext
WildcardedString : =
    "*" | String | {String "*"}+ [String]
    | ["*" String]+ ['*']

(String that is, an alternating sequence of String and "*")

String : =
    ":" Char*:"
    | "%s"

Ext : =
    ":name'(" WildcardedString ")"
    | ":reftype'(" Identifier ")"
    | ":addrtype'(" Identifier ")"

Identifier : =
    :" Char*:"
    | ":i"

Char : =
    an element of the Portable Character Set (ASCII)
    | a character in the repertoire of a string representation
```

**EXIT STATUS**

0  Operation was successful.

1  Operation failed.

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWfns</td>
</tr>
</tbody>
</table>

**SEE ALSO**

printf(1), FN_search_control_t(3XFN), FN_search_filter_t(3XFN),
fn_attr_ext_search(3XFN), fn_attr_search(3XFN), attributes(5), fns(5)

**NOTES**

If the filter expression is empty, it evaluates to TRUE (all objects satisfy it).

If the identifier in any subexpression of the filter expression does not exist as an attribute of an object, then the innermost logical expression containing that identifier evaluates to FALSE.
fnunbind(1)

NAME fnunbind – unbind the reference from an FNS name

SYNOPSIS fnunbind composite_name

DESCRIPTION fnunbind unbinds the reference of composite_name.

For example,

eg% fnunbind user/jsmith/fs/

unbinds the reference to which the name user/jsmith/fs/ was bound.

Note that an fnunbind on a name of a context will fail because such a context cannot be unbound without destroying it first with the command fndestroy.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWfns</td>
</tr>
</tbody>
</table>

SEE ALSO fnbind(1), fnlist(1), fnlookup(1), fnrename(1), fncreate(1M), fndestroy(1M), attributes(5), fns(5)
The `fold` utility is a filter that will fold lines from its input files, breaking the lines to have a maximum of `width` column positions (or bytes, if the `-b` option is specified). Lines will be broken by the insertion of a NEWLINE character such that each output line (referred to later in this section as a segment) is the maximum width possible that does not exceed the specified number of column positions (or bytes). A line will not be broken in the middle of a character. The behavior is undefined if `width` is less than the number of columns any single character in the input would occupy.

If the CARRIAGE-RETURN, BACKSPACE, or TAB characters are encountered in the input, and the `-b` option is not specified, they will be treated specially:

- **BACKSPACE**
  - The current count of line width will be decremented by one, although the count never will become negative.
  - `fold` will not insert a NEWLINE character immediately before or after any BACKSPACE character.

- **CARRIAGE-RETURN**
  - The current count of line width will be set to 0. `fold` will not insert a NEWLINE character immediately before or after any CARRIAGE-RETURN character.

- **TAB**
  - Each TAB character encountered will advance the column position pointer to the next tab stop. Tab stops will be at each column position \( n \) such that \( n \mod 8 = 1 \).

The following options are supported:

- `-b` Count `width` in bytes rather than column positions.
- `-s` If a segment of a line contains a blank character within the first `width` column positions (or bytes), break the line after the last such blank character meeting the width constraints. If there is no blank character meeting the requirements, the `-s` option will have no effect for that output segment of the input line.
- `-w width | -width` Specify the maximum line length, in column positions (or bytes if `-b` is specified). If `width` is not a positive decimal number, an error is returned. The default value is 80.

The following operand is supported:

- **file** A path name of a text file to be folded. If no `file` operands are specified, the standard input will be used.
EXAMPLE 1 Submitting a file of possibly long lines to the line printer.

An example invocation that submits a file of possibly long lines to the line printer (under the assumption that the user knows the line width of the printer to be assigned by `lp(1)`):

```bash
example% fold -w 132 bigfile | lp
```

See `environ(5)` for descriptions of the following environment variables that affect the execution of `fold`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

The following exit values are returned:

- `0` All input files were processed successfully.
- `>0` An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

`cut(1)`, `pr(1)`, `attributes(5)`, `environ(5)`

`fold(1)` can be used to create text files out of files with arbitrary line lengths. `fold` should be used when the contents of long lines need to be kept contiguous. `cut` should be used when the number of lines (or records) needs to remain constant.

`fold` is frequently used to send text files to line printers that truncate, rather than fold, lines wider than the printer is able to print (usually 80 or 132 column positions).

`fold` may not work correctly if underlining is present.
NAME  from – display the sender and date of newly-arrived mail messages

SYNOPSIS  /usr/ucb/from [-s sender] [username]

DESCRIPTION  The from utility prints out the mail header lines in your mailbox file to show you who your mail is from. If username is specified, then username’s mailbox is examined instead of your own.

OPTIONS  -s sender Only display headers for mail sent by sender.

USAGE  See largefile(5) for the description of the behavior of from when encountering files greater than or equal to 2 Gbyte (2^31 bytes).

FILES  /var/spool/mail/*

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

SEE ALSO  biff(1B), mail(1B), attributes(5), largefile(5)
ftp(1)

NAME

ftp – file transfer program

SYNOPSIS

ftp [-dginptv] [-T timeout] [hostname [port]]

DESCRIPTION

The ftp command is the user interface to the Internet standard File Transfer Protocol (FTP). ftp transfers files to and from a remote network site.

The host and optional port with which ftp is to communicate may be specified on the command line. If this is done, ftp immediately attempts to establish a connection to an FTP server on that host. Otherwise, ftp enters its command interpreter and awaits instructions from the user. When ftp is awaiting commands from the user, it displays the prompt ftp>.

OPTIONS

The following options may be specified at the command line, or to the command interpreter:

- d
  Enables debugging.

- g
  Disables filename “globbing”.

- i
  Turns off interactive prompting during multiple file transfers.

- n
  Does not attempt “auto-login” upon initial connection. If auto-login is not disabled, ftp checks the .netrc file in the user’s home directory for an entry describing an account on the remote machine. If no entry exists, ftp will prompt for the login name of the account on the remote machine (the default is the login name on the local machine), and, if necessary, prompts for a password and an account with which to login.

- p
  Enables passive mode for data transfers. This command is useful when connecting to a remote host from behind a connection filtering firewall.

- t
  Enables packet tracing (unimplemented).

- T timeout
  Enables global connection timer, specified in seconds (decimal). There is a timer for the control connection that is reset when anything is sent to the server and disabled while the client is prompting for user input. Another independent timer is used to monitor incoming or outgoing data connections.

- v
  Shows all responses from the remote server, as well as report on data transfer statistics. This is turned on by default if ftp is running interactively with its input coming from the user’s terminal.

The following commands can be specified to the command interpreter:

! [ command ]
  Runs command as a shell command on the local machine. If no command is given, invokes an interactive shell.
Executes the macro macro-name that was defined with the \texttt{macdef} command. Arguments are passed to the macro unglobbed.

Supplies a supplemental password required by a remote system for access to resources once a login has been successfully completed. If no argument is included, the user will be prompted for an account password in a non-echoing input mode.

Appends a local file to a file on the remote machine. If \texttt{remote-file} is not specified, the local file name is used, subject to alteration by any \texttt{ntrans} or \texttt{nmap} settings. File transfer uses the current settings for "representation type", "file structure", and "transfer mode".

Sets the "representation type" to "network ASCII". This is the default type.

Sounds a bell after each file transfer command is completed.

Sets the "representation type" to "image".

Terminates the FTP session with the remote server and exit \texttt{ftp}. An \texttt{EOF} will also terminate the session and exit.

Toggles remote computer file name case mapping during \texttt{mget} commands. When \texttt{case} is on (default is off), remote computer file names with all letters in upper case are written in the local directory with the letters mapped to lower case.

Changes the working directory on the remote machine to \texttt{remote-directory}.

Changes the remote machine working directory to the parent of the current remote machine working directory.

Terminates the FTP session with the remote server, and return to the command interpreter. Any defined macros are erased.

Toggles RETURN stripping during "network ASCII" type file retrieval. Records are denoted by a \texttt{RETURN/LINEFEED} sequence during "network ASCII" type file transfer. When \texttt{cr} is on (the default), RETURN characters are stripped from this sequence to conform with the UNIX system single \texttt{LINEFEED} record delimiter. Records on non-UNIX-system remote hosts may contain single \texttt{LINEFEED} characters; when an "network ASCII" type transfer is made, these \texttt{LINEFEED} characters may be distinguished from a record delimiter only when \texttt{cr} is off.
delete remote-file
   Deletes the file remote-file on the remote machine.

dump
glob
   Toggles debug printing. When debug is on, ftp prints each command sent to
   the remote machine, preceded by the string -->.

dump
   Toggles debugging mode. When debugging is on, ftp prints each command sent to
   the remote machine, preceded by the string -->.

dir [ remote-directory ] [ local-file ]
   Prints a listing of the directory contents in the directory, remote-directory, and,
   optionally, placing the output in local-file. If no directory is specified, the current
   working directory on the remote machine is used. If no local file is specified, or
   local-file is -, output is sent to the terminal.
dir
   A synonym for close.
form [ format-name ]
   Sets the carriage control format subtype of the “representation type” to format-name.
   The only valid format-name is non-print, which corresponds to the default
   “non-print” subtype.
get remote-file [ local-file ]
   Retrieves the remote-file and store it on the local machine. If the local file name is not
   specified, it is given the same name it has on the remote machine, subject to
   alteration by the current case, ntrans, and nmap settings. The current settings for
   “representation type”, “file structure”, and “transfer mode” are used while
   transferring the file.
glob
   Toggles filename expansion, or “globbing”, for mdelete, mget and mput. If
   globbing is turned off, filenames are taken literally.
   Globbing for mput is done as in sh(1). For mdelete and mget, each remote file
   name is expanded separately on the remote machine, and the lists are not merged.
   Expansion of a directory name is likely to be radically different from expansion of
   the name of an ordinary file: the exact result depends on the remote operating
   system and FTP server, and can be previewed with the command, mls remote-files -.
   mget and mput are not meant to transfer entire directory subtrees of files. You can
   do this by transferring a tar(1) archive of the subtree (using a “representation
   type” of “image” as set by the binary command).
hash
   Toggles hash-sign (#) printing for each data block transferred. The size of a data
   block is 8192 bytes.
help [ command ]
   Prints an informative message about the meaning of command. If no argument is
   given, ftp prints a list of the known commands.
Changes the working directory on the local machine. If no directory is specified, the user’s home directory is used.

Prints an abbreviated listing of the contents of a directory on the remote machine. If remote-directory is left unspeciﬁed, the current working directory is used.

The -a option lists all entries, including those that begin with a dot (.), which are normally not listed. The -l option lists ﬁles in long format, giving mode, number of links, owner, group, size in bytes, and time of last modiﬁcation for each ﬁle. If the ﬁle is a special ﬁle, the size ﬁeld instead contains the major and minor device numbers rather than a size. If the ﬁle is a symbolic link, the ﬁlename is printed followed by “→” and the pathname of the referenced ﬁle.

If no local ﬁle is speciﬁed, or if local-ﬁle is −, the output is sent to the terminal.

Defines a macro. Subsequent lines are stored as the macro macro-name. A null line (consecutive NEWLINE characters in a ﬁle or RETURN characters from the terminal) terminates macro input mode. There is a limit of 16 macros and 4096 total characters in all deﬁned macros. Macros remain deﬁned until a close command is executed.

The macro processor interprets $ and \ as special characters. A $ followed by a number (or numbers) is replaced by the corresponding argument on the macro invocation command line. A $ followed by an i signals that macro processor that the executing macro is to be looped. On the ﬁrst pass, $i is replaced by the ﬁrst argument on the macro invocation command line; on the second pass, it is replaced by the second argument, and so on. A \ followed by any character is replaced by that character. Use the \ to prevent special treatment of the $.

Deletes the remote-ﬁles on the remote machine.

Like dir, except multiple remote ﬁles may be speciﬁed. If interactive prompting is on, ftp will prompt the user to verify that the last argument is indeed the target local ﬁle for receiving mdir output.

Expands the remote-ﬁles on the remote machine and do a get for each ﬁle name thus produced. See glob for details on the ﬁlename expansion. Resulting ﬁle names will then be processed according to case, ntrans, and nmap settings. Files are transferred into the local working directory, which can be changed with lcd directory. New local directories can be created with ! mkdir directory.

Makes a directory on the remote machine.
m1s remote-files local-file
   Like ls(1), except multiple remote files may be specified. If interactive prompting is 
on, ftp will prompt the user to verify that the last argument is indeed the target 
local file for receiving m1s output.

mode [ mode-name ]
   Sets the “transfer mode” to mode-name. The only valid mode-name is stream, which 
corresponds to the default “stream” mode. This implementation only supports 
stream, and requires that it be specified.

mput local-files
   Expands wild cards in the list of local files given as arguments and do a put for 
each file in the resulting list. See glob for details of filename expansion. Resulting 
file names will then be processed according to ntrans and nmap settings.

nmap [ inpattern outpattern ]
   Sets or unsets the filename mapping mechanism. If no arguments are specified, the 
filename mapping mechanism is unset. If arguments are specified, remote filenames 
are mapped during mput commands and put commands issued without a 
specified remote target filename. If arguments are specified, local filenames are 
mapped during mget commands and get commands issued without a specified 
local target filename. 

   This command is useful when connecting to a non-UNIX-system remote host with 
different file naming conventions or practices. The mapping follows the pattern set 
by inpattern and outpattern. inpattern is a template for incoming filenames (which 
may have already been processed according to the ntrans and case settings). 
Variable templating is accomplished by including the sequences $1, $2, ..., $9 in 
inpattern. Use \ to prevent this special treatment of the $ character. All other 
characters are treated literally, and are used to determine the nmap inpattern 
variable values.

   For example, given inpattern $1.$2 and the remote file name mydata.data, $1 
would have the value mydata, and $2 would have the value data.

   The outpattern determines the resulting mapped filename. The sequences $1, $2, 
..., $9 are replaced by any value resulting from the inpattern template. The 
sequence $0 is replaced by the original filename. Additionally, the sequence 
[ seq1 , seq2 ] is replaced by seq1 if seq1 is not a null string; otherwise it is replaced by 
seq2.

   For example, the command nmap $1.$2.$3 [ $1,$2].[$2,file] would yield 
the output filename myfile.data for input filenames myfile.data and 
myfile.data.old, myfile.file for the input filename myfile, and 
myfile.myfile for the input filename .myfile. SPACE characters may be 
included in outpattern, as in the example nmap $1 | sed "s/ \*$//" > $1. Use 
the \ character to prevent special treatment of the $, [ ], and , characters.

ntrans [ inchars [ outchars ] ]
   Sets or unsets the filename character translation mechanism. If no arguments are 
specified, the filename character translation mechanism is unset. If arguments are
specified, characters in remote filenames are translated during mput commands and put commands issued without a specified remote target filename, and characters in local filenames are translated during mget commands and get commands issued without a specified local target filename.

This command is useful when connecting to a non-UNIX-system remote host with different file naming conventions or practices. Characters in a filename matching a character in inchars are replaced with the corresponding character in outchars. If the character’s position in inchars is longer than the length of outchars, the character is deleted from the file name.

Only 16 characters can be translated when using the ntrans command under ftp. Use case (described above) if needing to convert the entire alphabet.

open host [ port ]
Establishes a connection to the specified host FTP server. An optional port number may be supplied, in which case, ftp will attempt to contact an FTP server at that port. If the auto-login option is on (default setting), ftp will also attempt to automatically log the user in to the FTP server.

passive
Toggles passive mode. When passive mode is turned on, the ftp client sends the PASV command requesting that the FTP server open a port for the data connection and return the address of that port. The remote server listens on that port and the client connects to it. When passive mode is turned off, the ftp client sends the PORT command to the server specifying an address for the remove server to connect back to. Passive mode is useful when the connections to the ftp client are controlled, for example, when behind a firewall. When connecting to an IPv6–enabled FTP server, EPSV may be used in place of PASV and EPRT in place of PORT.

prompt
Toggles interactive prompting. Interactive prompting occurs during multiple file transfers to allow the user to selectively retrieve or store files. By default, prompting is turned on. If prompting is turned off, any mget or mput will transfer all files, and any mdelete will delete all files.

proxy ftp-command
Executes an FTP command on a secondary control connection. This command allows simultaneous connection to two remote FTP servers for transferring files between the two servers. The first proxy command should be an open, to establish the secondary control connection. Enter the command proxy ? to see other FTP commands executable on the secondary connection.

The following commands behave differently when prefaced by proxy: open will not define new macros during the auto-login process, close will not erase existing macro definitions, get and mget transfer files from the host on the primary control connection to the host on the secondary control connection, and put, mputd, and append transfer files from the host on the secondary control connection to the host on the primary control connection.
Third party file transfers depend upon support of the `PASV` command by the server on the secondary control connection.

**put** *local-file [ remote-file ]*

Stores a local file on the remote machine. If `remote-file` is left unspecified, the local file name is used after processing according to any `ntrans` or `nmap` settings in naming the remote file. File transfer uses the current settings for “representation type”, “file structure”, and “transfer mode”.

**pwd**

Prints the name of the current working directory on the remote machine.

**quit**

A synonym for `bye`.

**quote** *arg1 arg2 ...*

Sends the arguments specified, verbatim, to the remote FTP server. A single FTP reply code is expected in return. (The `remotehelp` command displays a list of valid arguments.)

quote should be used only by experienced users who are familiar with the FTP protocol.

**recv** *remote-file [ local-file ]*

A synonym for `get`.

**reget** *remote-file [ local-file ]*

The `reget` command acts like `get`, except that if `local-file` exists and is smaller than `remote-file`, `local-file` is presumed to be a partially transferred copy of `remote-file` and the transfer is continued from the apparent point of failure. This command is useful when transferring large files over networks that are prone to dropping connections.

**remotehelp** [ *command-name* ]

Requests help from the remote FTP server. If a `command-name` is specified it is supplied to the server as well.

**rename** *from to*

Renames the file `from` on the remote machine to have the name `to`.

**reset**

Clears reply queue. This command re-synchronizes command/reply sequencing with the remote FTP server. Resynchronization may be necessary following a violation of the FTP protocol by the remote server.

**restart** [ *marker* ]

Restarts the immediately following `get` or `put` at the indicated marker. On UNIX systems, `marker` is usually a byte offset into the file. When followed by an `mget`, the `restart` applies to the first `get` performed. Specifying a `marker` of 0 clears the restart marker. If no argument is specified, the current restart status is displayed.

**rmdir** *directory-name*

Deletes a directory on the remote machine.
runique
Toggles storing of files on the local system with unique filenames. If a file already
exists with a name equal to the target local filename for a get or mget command, a
.1 is appended to the name. If the resulting name matches another existing file, a
.2 is appended to the original name. If this process continues up to .99, an error
message is printed, and the transfer does not take place. The generated unique
filename will be reported. runique will not affect local files generated from a shell
command. The default value is off.

send local-file [ remote-file ]
A synonym for put.

sendport
Toggles the use of PORT commands. By default, ftp will attempt to use a PORT
command when establishing a connection for each data transfer. The use of PORT
commands can prevent delays when performing multiple file transfers. If the PORT
command fails, ftp will use the default data port. When the use of PORT
commands is disabled, no attempt will be made to use PORT commands for each
data transfer. This is useful when connected to certain FTP implementations that
ignore PORT commands but incorrectly indicate they have been accepted.

site arg1 [ arg2 ] ...  
Sends the arguments specified, verbatim, to the remote FTP server as a SITE
command.

status
Show the current status of ftp.

struct [ struct-name ]
Sets the file structure to struct-name. The only valid struct-name is file, which
corresponds to the default “file” structure. The implementation only supports
file, and requires that it be specified.

sunique
Toggles storing of files on remote machine under unique file names. The remote
FTP server must support the STOU command for successful completion. The remote
server will report the unique name. Default value is off.

tcpwindow [ size ]
Sets the TCP window size to be used for data connections. Specifying a size of 0
stops the explicit setting of the TCP window size on data connections. If no
argument is specified, the current setting is displayed.

tenex
Sets the “representation type” to that needed to talk to TENEX machines.

trace
Toggles packet tracing (unimplemented).

type [ type-name ]
Sets the “representation type” to type-name. The valid type-names are ascii for
“network ASCII”, binary or image for “image”, and tenex for “local byte size”
ftp(1)

with a byte size of 8 (used to talk to TENEX machines). If no type is specified, the current type is printed. The default type is “network ASCII”.

```
user user-name [ password ] [ account ]
```

Identify yourself to the remote FTP server. If the password is not specified and the server requires it, `ftp` will prompt the user for it (after disabling local echo). If an account field is not specified, and the FTP server requires it, the user will be prompted for it. If an account field is specified, an account command will be relayed to the remote server after the login sequence is completed if the remote server did not require it for logging in. Unless `ftp` is invoked with “auto-login” disabled, this process is done automatically on initial connection to the FTP server.

```
verbose
```

Toggles verbose mode. In verbose mode, all responses from the FTP server are displayed to the user. In addition, if verbose mode is on, when a file transfer completes, statistics regarding the efficiency of the transfer are reported. By default, verbose mode is on if `ftp`’s commands are coming from a terminal, and off otherwise.

```
? [ command ]
```

A synonym for `help`.

Command arguments which have embedded spaces may be quoted with quote (") marks.

If any command argument which is not indicated as being optional is not specified, `ftp` will prompt for that argument.

**ABORTING A FILE TRANSFER**

To abort a file transfer, use the terminal interrupt key. Sending transfers will be immediately halted. Receiving transfers will be halted by sending an FTP protocol `ABOR` command to the remote server, and discarding any further data received. The speed at which this is accomplished depends upon the remote server’s support for `ABOR` processing. If the remote server does not support the `ABOR` command, an `ftp>` prompt will not appear until the remote server has completed sending the requested file.

The terminal interrupt key sequence will be ignored when `ftp` has completed any local processing and is awaiting a reply from the remote server. A long delay in this mode may result from the `ABOR` processing described above, or from unexpected behavior by the remote server, including violations of the ftp protocol. If the delay results from unexpected remote server behavior, the local `ftp` program must be killed by hand.

**FILE NAMING CONVENTIONS**

Local files specified as arguments to `ftp` commands are processed according to the following rules.

1) If the file name – is specified, the standard input (for reading) or standard output (for writing) is used.

2) If the first character of the file name is |, the remainder of the argument is interpreted as a shell command. `ftp` then forks a shell, using `popen(3C)`
with the argument supplied, and reads (writes) from the standard output (standard input) of that shell. If the shell command includes SPACE characters, the argument must be quoted; for example " | ls -lt". A particularly useful example of this mechanism is: "dir | more".

3) Failing the above checks, if globbing is enabled, local file names are expanded according to the rules used in the sh(1); see the glob command. If the ftp command expects a single local file (for example, put), only the first filename generated by the globbing operation is used.

4) For mget commands and get commands with unspecified local file names, the local filename is the remote filename, which may be altered by a case, ntrans, or nmap setting. The resulting filename may then be altered if runique is on.

5) For mput commands and put commands with unspecified remote file names, the remote filename is the local filename, which may be altered by a ntrans or nmap setting. The resulting filename may then be altered by the remote server if sunique is on.

The FTP specification specifies many parameters which may affect a file transfer.

The “representation type” may be one of “network ASCII”, “EBCDIC”, “image”, or “local byte size” with a specified byte size (for PDP-10’s and PDP-20’s mostly). The “network ASCII” and “EBCDIC” types have a further subtype which specifies whether vertical format control (NEWLINE characters, form feeds, etc.) are to be passed through (“non-print”), provided in TELNET format (“TELNET format controls”), or provided in ASA (FORTRAN) (“carriage control (ASA)”) format. ftp supports the “network ASCII” (subtype “non-print” only) and “image” types, plus “local byte size” with a byte size of 8 for communicating with TENEX machines.

The “file structure” may be one of file (no record structure), record, or page. ftp supports only the default value, which is file.

The “transfer mode” may be one of stream, block, or compressed. ftp supports only the default value, which is stream.

See largefile(5) for the description of the behavior of ftp when encountering files greater than or equal to 2 Gbyte (2³¹ bytes).

The ftp command is IPv6-enabled. See ip6(7P).

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWbip</td>
</tr>
</tbody>
</table>
ATTRIBUTE TYPE | ATTRIBUTE VALUE
--- | ---
CSI | enabled

### SEE ALSO
ls(1), rcp(1), sh(1), tar(1), in.ftpd(1M), popen(3C), ftpusers(4), netrc(4), attributes(5), largefile(5), ip6(7P)


### NOTES
Failure to log in may arise from an explicit denial by the remote FTP server because the account is listed in /etc/ftpusers. See in.ftpd(1M) and ftpusers(4).

Correct execution of many commands depends upon proper behavior by the remote server.

An error in the treatment of carriage returns in the 4.2 BSD code handling transfers with a “representation type” of “network ASCII” has been corrected. This correction may result in incorrect transfers of binary files to and from 4.2 BSD servers using a “representation type” of “network ASCII”. Avoid this problem by using the “image” type.
NAME  ftpcount – show current number of users in each FTP Server class

SYNOPSIS  ftpcount [-V]

DESCRIPTION  Use the ftpcount command to show the current number of users logged in and the login limit for each FTP Server class defined in the ftpaccess(4) file.

OPTIONS  The ftpcount command supports the following options:

-  `[-V]`  Display program copyright and version information, then terminate.

EXIT STATUS  The following exit values are returned:

0  Successful completion.

>0  An error occurred.

FILES  
/var/run/ftp.pids-classnames
/etc/ftpd/ftpaccess

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWftpui</td>
</tr>
</tbody>
</table>

SEE ALSO  ftpwho(1), in.ftpd(1M), ftpaccess(4), attributes(5)
ftpwho(1)

NAME
ftpwho – show current process information for each FTP Server user

SYNOPSIS
ftpwho [-V]

DESCRIPTION
Use the ftpwho command to show the current process information for each user
logged in to the FTP Server. This information is in addition to information displayed
by the ftpcount(1) command.

OPTIONS
The ftpwho command supports the following options:

- V
  Display the program copyright and version information, then terminate.

EXIT STATUS
The following exit values are returned:

0  Successful completion.

>0  An error occurred.

FILES
/etc/ftpd/ftpaccess
/var/run/ftp.pids-classname

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWftpu</td>
</tr>
</tbody>
</table>

SEE ALSO
ftpcount(1), ps(1), in.ftpd(1M), ftpaccess(4), attributes(5)
NAME
gcore – get core images of running processes

SYNOPSIS
gcore [-o filename] process-id...

DESCRIPTION
The gcore utility creates a core image of each specified process. By default, the name of the core image file for the process whose process ID is process-id will be core processo-id.

OPTIONS
-o filename Substitutes filename in place of core as the first part of the name of the core image files.

OPERANDS
process-id process ID

EXIT STATUS
0 On success.
non-zero On failure, such as non-existent process ID.

FILES
core .process-id core images

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWtwo (32-bit)</td>
</tr>
<tr>
<td></td>
<td>SUNWtwox (64-bit)</td>
</tr>
</tbody>
</table>

SEE ALSO
kill(1), core(4), proc(4), attributes(5)
The `gencat` command merges the message text source file(s) `msgfile` into a formatted message database `catfile`. The database `catfile` is created if it does not already exist. If `catfile` does exist, its messages are included in the new `catfile`. If set and message numbers collide, the new message-text defined in `msgfile` replaces the old message text currently contained in `catfile`. The message text source file (or set of files) input to `gencat` can contain either set and message numbers or simply message numbers, in which case the set `NL_SETD` (see `nl_types(3HEAD)`) is assumed.

The format of a message text source file is defined as follows. Note that the fields of a message text source line are separated by a single ASCII space or tab character. Any other ASCII spaces or tabs are considered as part of the subsequent field.

- **$set n comment**
  Where `n` specifies the set identifier of the following messages until the next `$set`, `$delset`, or end-of-file appears. `n` must be a number in the range `(1–{NL_SETMAX})`. Set identifiers within a single source file need not be contiguous. Any string following the set identifier is treated as a comment. If no `$set` directive is specified in a message text source file, all messages are located in the default message set `NL_SETD`.

- **$delset n comment**
  Deletes message set `n` from an existing message catalog. Any string following the set number is treated as a comment. (Note: if `n` is not a valid set it is ignored.)

- **$comment**
  A line beginning with a dollar symbol `$` followed by an ASCII space or tab character is treated as a comment.

- **m message-text**
  The `m` denotes the message identifier, a number in the range `(1–{NL_MSGMAX})`. The `message-text` is stored in the message catalog with the set identifier specified by the last `$set` directive, and with message identifier `m`. If the `message-text` is empty, and an ASCII space or tab field separator is present, an empty string is stored in the message catalog. If a message source line has a message number, but neither a field separator nor `message-text`, the existing message with that number (if any) is deleted from the catalog. Message identifiers need not be contiguous. The length of `message-text` must be in the range `(0–{NL_TEXTMAX})`.

- **$quote c**
  This line specifies an optional quote character `c`, which can be used to surround `message-text` so that trailing spaces or null (empty) messages are visible in a message source line. By default, or if an empty `$quote`
directive is supplied, no quoting of message-text will be recognized.

Empty lines in a message text source file are ignored.

Text strings can contain the special characters and escape sequences defined in the following table:

<table>
<thead>
<tr>
<th>Description</th>
<th>Symbol</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>newline</td>
<td>NL(LF)</td>
<td>\n</td>
</tr>
<tr>
<td>horizontal tab</td>
<td>HT</td>
<td>\t</td>
</tr>
<tr>
<td>vertical tab</td>
<td>VT</td>
<td>\v</td>
</tr>
<tr>
<td>backspace</td>
<td>BS</td>
<td>\b</td>
</tr>
<tr>
<td>carriage return</td>
<td>CR</td>
<td>\r</td>
</tr>
<tr>
<td>form feed</td>
<td>FF</td>
<td>\f</td>
</tr>
<tr>
<td>backlash</td>
<td>\</td>
<td>\</td>
</tr>
<tr>
<td>bit pattern</td>
<td>ddd</td>
<td>\ddd</td>
</tr>
</tbody>
</table>

The escape sequence \ddd consists of backslash followed by 1, 2 or 3 octal digits, which are taken to specify the value of the desired character. If the character following a backslash is not one of those specified, the backslash is ignored.

Backslash followed by an ASCII newline character is also used to continue a string on the following line. Thus, the following two lines describe a single message string:

```
1 This line continues \n to the next line
```

which is equivalent to:

```
1 This line continues to the next line
```

**OPERANDS**

The following operands are supported:

- **catfile**
  A path name of the formatted message catalogue. If – is specified, standard output is used.

- **msgfile**
  A path name of a message text source file. If – is specified for an instance of msgfile, standard input is used. The format of message text source files is defined in Message Text Source File Format.
See environ(5) for descriptions of the following environment variables that affect the execution of gencat: LC_CTYPE, LC_MESSAGES, and NLSPATH.

The following exit values are returned:

- 0    Successful completion.
- >0   An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWloc</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

SEE ALSO
mkmsgs(1), catgets(3C), catopen(3C),gettext(3C), environ(5), attributes(5), nl_types(3HEAD)
NAME
geniconvtbl – generate iconv code conversion tables

SYNOPSIS
            [-Idirectory] [-Uname] [infile...]

DESCRIPTION
The geniconvtbl utility accepts code conversion rules defined in flat text file(s) and
writes code conversion binary table file(s) that can be used to support user-defined
iconv code conversions (see iconv(1) and iconv(3C) for more detail on the iconv
code conversion).

OPTIONS
The following options are supported:

-\f
  Overwrites output file if the output file exists.

-\n
  Does not generate an output file. This is useful to check
  the contents of the input file.

-\p preprocessor
  Uses specified preprocessor instead of the default
  preprocessor, /usr/lib/cpp.

-\q
  Quiet option. It suppresses warning and error
  messages.

-\W arg
  Passes the argument arg to the preprocessor. If this
  option is specified more than once, all arguments are
  passed to the preprocessor.

-\D name
-\D name=def
-\Idirectory
-\U name
  geniconvtbl recognizes these options and passes
  them and their arguments to the preprocessor.

OPERANDS
The following operand is supported:

infile
  A path name of an input file. If no input file is
  specified, geniconvtbl reads from the standard input
  stream. The user can specify more than one input file if
  necessary.

OUTPUT
If input is from the standard input stream, geniconvtbl writes output to the
standard output stream. If one or more input files are specified, geniconvtbl reads
from each input file and writes to a corresponding output file. Each of the output file
names will be the same as the corresponding input file with .bt appended.

The generated output files must be moved to the following directory prior to using the
code conversions at iconv(1) and iconv(3C):

/usr/lib/iconv/geniconvtbl/binarytables/The output file name should start with one or more printable ASCII characters as the ‘fromcode’ name followed
by a percentage character (%), followed by one or more printable ASCII characters as
the ‘tocode’ name, followed by the suffix ‘.bt’. The ‘fromcode’ and ‘tocode’


names are used to identify the `iconv` code conversion at `iconv(1)` and `iconv_open(3C)`). The properly named output file should be placed in the directory, `/usr/lib/iconv/geniconvtbl/binarytables/.

**EXAMPLE 1** Generating an `iconv` code conversion binary table

The following example generates a code conversion binary table with output file name `convertA2B.bt`:

```bash
example@ geniconvtbl convertA2B
```

**EXAMPLE 2** Generating multiple `iconv` code conversion binary tables

The following example generates two code conversion binary tables with output files `test1.bt` and `test2.bt`:

```bash
example@ geniconvtbl test1 test2
```

**EXAMPLE 3** Using another preprocessor

The following example generates a code conversion binary table once the specified preprocessor has processed the input file:

```bash
example@ geniconvtbl -p /opt/SUNWspro/bin/cc -W -E convertB2A
```

**EXAMPLE 4** Placing a binary table

To use the binary table created in the first example above as the engine of the conversion ‘fromcode’ ABC to ‘tocode’ DEF, become super-user and then rename it and place it like this:

```bash
example# mv convertA2B.bt \\
/usr/lib/iconv/geniconvtbl/binarytables/ABC%DEF.bt
```

**EXAMPLE 5** Providing modified ISO8859-1 to UTF-8 code conversion

Write a `geniconvtbl` source file that defines the code conversion. For instance, you can copy over `/usr/lib/iconv/geniconvtbl/srcs/ISO8859-1_to_UTF-8.src` into your directory and make necessary changes at the source file. Once the modifications are done, generate the binary table:

```bash
example@ geniconvtbl ISO8859-1_to_UTF-8.src
```

As super-user, place the generated binary table with a unique name at the system directory where `iconv_open(3C)` can find the binary table:

```bash
example su
Password:
example@ cp ISO8859-1_to_UTF-8.bt \\
/usr/lib/iconv/geniconvtbl/binarytables/my-iso-8859-1%utf-8.bt
```
EXAMPLE 5  Providing modified ISO8859-1 to UTF-8 code conversion  (Continued)

After that, you can do the iconv code conversion. For instance:

```bash
example> iconv -f my-iso-8859-1 -t utf-8 testfile.txt
```

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of geniconvtbl: LANG and LCCTYPE.

EXIT STATUS

The following exit values are returned:

0   No errors occurred and the output files were successfully created.
1   Command line options are not correctly used or an unknown command line option was specified.
2   Invalid input or output file was specified.
3   Conversion rules in input files are not correctly defined.
4   Conversion rule limit of input files has been reached. See NOTES section of geniconvtbl(4).
5   No more system resource error.
6   Internal error.

FILES

/usr/lib/iconv/geniconvtbl/binarytables/*.bt
conversion binary tables

/usr/lib/iconv/geniconvtbl/srcs/*
conversion source files for user reference

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO

cpp(1), iconv(1), iconv(3C), iconv_close(3C), iconv_open(3C), geniconvtbl(4), attributes(5), environ(5), iconv(5)

Solaris Internationalization Guide for Developers

NOTES

The generated and correctly placed output files, /usr/lib/iconv/geniconvtbl/binarytables/*.bt, are used in both 32-bit and 64-bit environments.
genlayouttbl(1)

NAME  
genlayouttbl – generate layout table for complex text layout

SYNOPSIS  
genlayouttbl [ -o outfile ] [ infile ]

DESCRIPTION  
The genlayouttbl utility accepts a locale’s layout definition in a flat text file and writes a binary layout table file that can be used in the complex text layout of the locale.

OPTIONS  
The following option is supported:
- o outfile  
  Writes output binary layout table to the outfile.

OPERANDS  
The following operand is supported:
  infile  
  A path name of an input file. If no input file is specified, genlayouttbl reads from the standard input stream.

OUTPUT AND SYMBOLIC LINKS  
If no outfile is specified, genlayouttbl writes output to the standard output stream.

The generated output file must be moved to the following directory prior to the use at the system and the file name should be layout.dat:

/usr/lib/locale/LO_LTYPE/layout.dat

The locale should also have a symbolic link,
/usr/lib/locale/LO_LTYPE/locale. layout.so.1, to the 32-bit Universal Multiscript Layout Engine (UMLE),
/usr/lib/locale/common/LO_LTYPE/umle.layout.so.1.

For proper 64-bit platform operations, the locale should also have a symbolic link, as for instance, in 64-bit SPARC platform,

The locale is the locale that you want to provide and to use the layout functionality you defined.

INPUT FILE FORMAT  
A layout definition file to genlayouttbl contains three different sections of definitions:
- Layout attribute definition
- Bidirectional data and character type data definition
- Shaping data definition

For appropriate complex text layout support, all three sections need to be defined in the layout definition file.

The Lexical Conventions  
The following lexical conventions are used in the layout definition:
NAME  A string of characters that consists of printable ASCII characters. It includes DECIMAL and HEXADECIMAL also. *Examples:* `test, a1_src, b32, 123`.

HEXADECIMAL_BYTE  Two-digit hexadecimal number. The number starts with a hexadecimal digit followed by another hexadecimal digit. *Examples:* `e0, E1, a7, fa`.

HEXADECIMAL  A hexadecimal number. The hexadecimal representation consists of an escape character, '0' followed by the constant 'x' or 'X' and one or more hexadecimal digits. *Examples:* `0x0, 0x1, 0x1a, 0xA, 0x1b3`.

DECIMAL  A decimal number, represented by one or more decimal digits. *Examples:* `0, 123, 2165`.

Each comment must start with '#'. The comment ends at the end of the line.

The following keywords are reserved:

- active_directional, active_shape_editing, AL,
- ALGORITHM_BASIC, ALGORITHM_IMPLICIT, AN, BN, check_mode,
- context, CONTEXT_LTR, CONTEXT_RTL, CS, EN, END, ES, ET, FALSE,
- FILE_CODE_REPRESENTATION, implicit_algorithm, keep, L,
- LAYOUT_ATTRIBUTES, LAYOUT_BIDI_CHAR_TYPE_DATA,
- LAYOUT_SHAPE_DATA, LRE, LRO, MODE_EDIT, MODE_STREAM, NSM,
- national_numerals, numerals, NUMERALS_CONTEXTUAL,
- NUMERALS_NATIONAL, NUMERALS_NOMINAL, ON, orientation,
- ORIENTATION_CONTEXTUAL, ORIENTATION_LTR, ORIENTATION_RTL,
- ORIENTATION_TTBRL, PDF,
- PROCESS_CODE_REPRESENTATION, PS, R, repeat*, repeat+, RLE, RLO, S,
- shape_charset, shape_charset_size, shape_context_size, swapping,
- SWAPPING_NO, swapping_pairs, SWAPPING_YES, TEXT_EXPLICIT,
- TEXT_IMPLICIT, TEXT_NOMINAL, TEXT_SHAPE, text_shaping, TEXT_VISUAL,
- TRUE, type_of_text, WS

Additionally, the following symbols are also reserved as tokens:

```
(  ) [  ] ,  ;  . . . =  ->  +
```

The layout attribute definition section defines the layout attributes and their associated values.

The definition starts with a keyword, LAYOUT_ATTRIBUTES, and ends with END LAYOUT_ATTRIBUTES:

```plaintext
LAYOUT_ATTRIBUTES

# Layout attributes here.
```

User Commands  461
There are a total of eight layout attribute value trios that can be defined in this section:

- orientation
- context
- type_of_text
- implicit_algorithm
- swapping
- numerals
- text_shaping
- shape_context_size

Additionally, there are five layout attribute value pairs that also can be defined in this section:

- active_directional
- active_shape_editing
- shape_charset
- shape_charset_size
- check_mode

Each attribute value trio will have an attribute name, an attribute value for the input buffer, and an attribute value for the output buffer, as in the following example:

```plaintext
# Orientation layout attribute value trio. The input and output
# attribute values are separated by a colon and the left one
# is the input attribute value:
orientation ORIENTATION_LTR:ORIENTATION_LTR
```

Each attribute value pair will have an attribute name and an associated attribute value, as in the following example:

```plaintext
# Shape charset attribute value pair:
shape_charset ISO8859-6
```

The `orientation` value trio defines the global directional text orientation. The possible values are:

- **ORIENTATION_LTR**: Left-to-right horizontal rows that progress from top to bottom.
- **ORIENTATION_RTL**: Right-to-left horizontal rows that progress from top to bottom.
- **ORIENTATION_TTBRL**: Top-to-bottom vertical columns that progress from right to left.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORIENTATION_TTBLR</td>
<td>Top-to-bottom vertical columns that progress from left to right.</td>
</tr>
<tr>
<td>ORIENTATION_CONTEXTUAL</td>
<td>The global orientation is set according to the direction of the first significant (strong) character. If there are no strong characters in the text and the attribute is set to this value, the global orientation of the text is set according to the value of the attribute context. This value is meaningful only for bidirectional text.</td>
</tr>
</tbody>
</table>

If no value or value trio is defined, the default is ORIENTATION_LTR.

The context value trio is meaningful only if the attribute orientation is set to ORIENTATION_CONTEXTUAL. It defines what orientation is assumed when no strong character appears in the text. The possible values are:

- **CONTEXT_LTR**: In the absence of characters with strong directionality in the text, orientation is assumed to be left-to-right rows progressing from top to bottom.
- **CONTEXT_RTL**: In the absence of characters with strong directionality in the text, orientation is assumed to be right-to-left rows progressing from top to bottom.

If no value or value trio is specified, the default is CONTEXT_LTR.

The type_of_text value trio specifies the ordering of the directional text. The possible values are:

- **TEXT_VISUAL**: Code elements are provided in visually ordered segments, which can be rendered without any segment inversion.
- **TEXT_IMPLICIT**: Code elements are provided in logically ordered segments. Logically ordered means that the order in which the characters are provided is the same as the order in which the characters are pronounced when reading the presented text or the order in which characters would be entered from a keyboard.
- **TEXT_EXPLICIT**: Code elements are provided in logically ordered segments with a set of embedded controls. Some examples of such embedded controls from ISO/IEC 10646-1 are:
  
  - LEFT-TO-RIGHT EMBEDDING (LRE)
  - RIGHT-TO-LEFT EMBEDDING (RLE)
  - RIGHT-TO-LEFT OVERRIDE (RLO)
  - LEFT-TO-RIGHT OVERRIDE (LRO)
  - POP DIRECTIONAL FORMAT (PDF)
If no value or value trio is specified, the default is TEXT_IMPLICIT.

The implicit_algorithm value trio specifies the type of bidirectional implicit algorithm used in reordering and shaping of directional or context-dependent text. The possible values are:

ALGORITHM_IMPLICIT  Directional code elements will be reordered using an implementation-defined implicit algorithm.
ALGORITHM_BASIC  Directional code elements will be reordered using a basic implicit algorithm defined in the Unicode standard.

Even though we allow two different values for the implicit_algorithm, since the Solaris implementation-defined implicit algorithm is based on the Unicode standard, there is no difference in behavior whether you choose ALGORITHM_IMPLICIT or ALGORITHM_BASIC for this attribute.

The default value is ALGORITHM_IMPLICIT.

The swapping value trio specifies whether symmetric swapping is applied to the text. The possible values are:

SWAPPING_YES  The text conforms to symmetric swapping.
SWAPPING_NO  The text does not conform to symmetric swapping.

If no value or value trio is specified, the default is SWAPPING_NO.

The numerals value trio specifies the shaping of numerals. The possible values are:

NUMERALS_NOMINAL  Nominal shaping of numerals using the Arabic numbers of the portable character set (in Solaris, ASCII digits).
NUMERALS_NATIONAL  National shaping of numerals based on the script of the locale. For instance, Thai digits in the Thai locale.
NUMERALS_CONTEXTUAL  Contextual shaping of numerals depending on the context script of surrounding text, such as Hindi numbers in Arabic text and Arabic numbers otherwise.

If no value or value trio is specified, the default is NUMERALS_NOMINAL.

The text_shaping value trio specifies the shaping; that is, choosing (or composing) the correct shape of the input or output text. The possible values are:

TEXT_SHADED  The text has presentation form shapes.
TEXT_NOMINAL  The text is in basic form.

If no value or value trio is specified, the default is TEXT_NOMINAL for input and TEXT_SHADED for output.
The `shape_context_size` value trio specifies the size of the context (surrounding code elements) that must be accounted for when performing active shape editing. If not defined, the default value 0 is used for the number of surrounding code elements at both front and rear:

```plaintext
# The shape_context_size for both front and rear surrounding code elements are all zero:
shape_context_size 0:0
```

The front and rear attribute values are separated by a colon, with the front value to the left of the colon.

The `active_directional` value pair specifies whether the current locale requires (bi-)directional processing. The possible values are:

- **TRUE**: Requires (bi-)directional processing.
- **FALSE**: Does not require (bi-)directional processing.

The `active_shape_editing` value pair specifies whether the current locale requires context-dependent shaping for presentation. The possible values are:

- **TRUE**: Requires context-dependent shaping.
- **FALSE**: Does not require context-dependent shaping.

The `shape_charset` value pair specifies the current locale’s shape charset on which the complex text layout is based. There are two different kinds of shape charset values that can be specified:

- A single shape charset
- Multiple shape charsets

For a single shape charset, it can be defined by using `NAME` as defined in the Lexical Convention section above. For multiple shape charsets, however, it should follow the syntax given below in extended BNF form:

```
multiple_shape_charset : charset_list ;

charset_list : charset | charset_list ; charset

charset : charset_name = charset_id ;

charset_name : NAME ;

charset_id : HEXADECIMAL_BYTE ;
```
For instance, the following is a valid multiple shape charsets value for the
shape_charset attribute:

```
# Multi-shape charsets:
shape_charset tis620.2533=e4;iso8859-8=e5;iso8859-6=e6
```

The shape_charset must be specified.

The shape_charset_size value pair specifies the encoding size of the current
shape_charset. The valid value is a positive integer from 1 to 4. If the multiple
shape charsets value is defined for the shape_charset attribute, the
shape_charset_size must be 4.

The shape_charset_size must be specified.

The check_mode value pair specifies the level of checking of the elements in the
input buffer for shaping and reordering purposes. The possible values are:

- **MODE_STREAM** The string in the input buffer is expected to have valid
  combinations of characters or character elements.

- **MODE_EDIT** The shaping of input text may vary depending on locale-specific
  validation or assumption.

When no value or value pair is not specified, the default value is **MODE_STREAM**.

This section defines the bidirectional and other character types that will be used in the
Unicode Bidirectional Algorithm and the shaping algorithm part of the UMLE.

The definition starts with a keyword **LAYOUT_BIDI_CHAR_TYPE_DATA** and ends
with **END LAYOUT_BIDI_CHAR_TYPE_DATA**:

```
LAYOUT_BIDI_CHAR_TYPE_DATA
    # Layout bidi definitions here.
    ;
    ;
END LAYOUT_BIDI_CHAR_TYPE_DATA
```

The bidirectional data and character type data definition should be defined for the two
different kinds of text shape forms, **TEXT_SHAPED** and **TEXT_NOMINAL**, depending on
the text_shaping attribute value and also for the two different kinds of text
representations, file code representation and process code representation (that is, wide
character representation):

```
LAYOUT_BIDI_CHAR_TYPE_DATA
    FILE_CODE_REPRESENTATION
    TEXT_SHAPED
    # TEXT_SHAPED bidi and character type data
```
Each bidi and character type data definition can have the following definitions:

- Bidirectional data type definition
- `swapping_pairs` character type definition
- `national_numerals` character type definition

There are nineteen different bidirectional data types that can be defined, as in the following table:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>Strong</td>
<td>Left-to-right</td>
</tr>
<tr>
<td>LRE</td>
<td>Strong</td>
<td>Left-to-right embedding</td>
</tr>
<tr>
<td>Keyword</td>
<td>Category</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>LRO</td>
<td>Strong</td>
<td>Left-to-right override</td>
</tr>
<tr>
<td>R</td>
<td>Strong</td>
<td>Right-to-left</td>
</tr>
<tr>
<td>AL</td>
<td>Strong</td>
<td>Right-to-left</td>
</tr>
<tr>
<td>RLE</td>
<td>Strong</td>
<td>Right-to-left embedding</td>
</tr>
<tr>
<td>RLO</td>
<td>Strong</td>
<td>Right-to-left override</td>
</tr>
<tr>
<td>PDF</td>
<td>Weak</td>
<td>Pop directional format</td>
</tr>
<tr>
<td>EN</td>
<td>Weak</td>
<td>European number</td>
</tr>
<tr>
<td>ES</td>
<td>Weak</td>
<td>European number separator</td>
</tr>
<tr>
<td>ET</td>
<td>Weak</td>
<td>European number terminator</td>
</tr>
<tr>
<td>AN</td>
<td>Weak</td>
<td>Arabic number</td>
</tr>
<tr>
<td>CS</td>
<td>Weak</td>
<td>Common number separator</td>
</tr>
<tr>
<td>PS</td>
<td>Separator</td>
<td>Paragraph separator</td>
</tr>
<tr>
<td>S</td>
<td>Separator</td>
<td>Segment separator</td>
</tr>
<tr>
<td>WS</td>
<td>Neutral</td>
<td>White space</td>
</tr>
<tr>
<td>ON</td>
<td>Neutral</td>
<td>Other neutrals</td>
</tr>
<tr>
<td>NSM</td>
<td>Weak</td>
<td>Non-spacing mark</td>
</tr>
<tr>
<td>BN</td>
<td>Weak</td>
<td>Boundary neutral</td>
</tr>
</tbody>
</table>

If not defined in this section, the characters belong to the other neutrals type, ON.

Each keyword list above will be accompanied by one or more hexadecimal ranges of characters that belong to the bidirectional character type. The syntax is as follows:

```plaintext
bidi_char_type : bidi_keyword ':' range_list;

bidi_keyword : 'L' |
               'LRE' |
               'LRO' |
               'R' |
               'AL' |
               'RLE' |
               'RLO' |
               'PDF' |
               'EN' |
               'ES' |
               'ET' |
               'AN' |
               'CS' |
               'PS' |
```
For example:

# Bidi character type definitions:
L: 0x26, 0x41...0x5a, 0xc380...0xc396, 0xe285a0...0xe28682
WS: 0x20, 0xc2a0, 0xe28080...0xe28086

The swapping_pairs specifies the list of swappable characters if SWAPPING_YES is specified as a value at the swapping value trio. The syntax of the swapping_pairs is as follows:

swapping_pair_list : swapping_keyword ':', swap_pair_list

swapping_keyword : 'swapping_pairs'

swap_pair_list : swap_pair
| swap_pair_list ':', swap_pair

swap_pair : '(' HEXADECIMAL ',', HEXADECIMAL ')'

For example:

# Swapping pair definitions:
swapping_pairs: (0x28, 0x29), (0x7b, 0x7d)

The national_numerals specifies the list of national digits that can be converted as the numerals value trio specifies. The syntax of the national_numerals is as follows:

numerals_list : numerals_keyword ':',
| numerals_list ':', contextual_range_list

numerals_keyword : 'national_numerals'
For instance:

# National numerals definition. The national number that will
# replace Arabic number 0 to 9 is 0, 0x41, 0x42, and so on.
# The contextual surrounding characters are 0x20 to 0x40 and
# 0x50 to 0x7f:
national_numerals:
  {0x0, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x49}
  ; 0x20...0x40, 0x50...0x7f

Unless NUMERALS_CONTEXTUAL is the value of the numerals attribute, the contextual
range list definition is meaningless.
The shaping data definition section defines the context-dependent shaping rules that will be used in the shaping algorithm of the UMLE.

The definition starts with a keyword, LAYOUT_SHAPE_DATA, and ends with END LAYOUT_SHAPE_DATA:

```plaintext
LAYOUT_SHAPE_DATA

# Layout shaping data definitions here.
:
:

END LAYOUT_SHAPE_DATA
```

The shaping data definition should be defined for the two different kinds of text shape forms, TEXT_SHAPED and TEXT_NOMINAL, depending on the text_shaping attribute value and also for the two different kinds of text representations, file code representation and process code representation (that is, wide character representation):

```plaintext
LAYOUT_SHAPE_DATA

FILE_CODE_REPRESENTATION
TEXT_SHAPED

# TEXT_SHAPED shaping data definition in file code representation here.
:
:

END TEXT_SHAPED

TEXT_NOMINAL

# TEXT_NOMINAL shaping data definition in file code representation here.
:
:

END TEXT_NOMINAL
END FILE_CODE_REPRESENTATION

PROCESS_CODE_REPRESENTATION
TEXT_SHAPED

# TEXT_SHAPED shaping data definition in process code representation here.
:
:

END TEXT_SHAPED

TEXT_NOMINAL

# TEXT_NOMINAL shaping data definition in process code representation here.
:
:
```

User Commands 471
Each shaping data definition consists of one or more of the shaping sequence definitions. Each shaping sequence definition is a representation of a series of state transitions triggered by an input character and the current state at each transition.

The syntax of the shaping sequence definition is as follows:

```
shaping_sequence : initial_state '+' input '->' next_state_list ;
initial_state : '{'()}';
input : HEXADECIMAL ;
next_state_list : next_state |
next_state_list '+' input '->' next_state |
'(' next_state_list '+' input ')','repeat*'
| '>(' next_state_list '+' input ')','repeat*'
;
next_state : '('out_buffer ','in2out ','out2in ','property ')';
out_buffer : '['out_char_list ']';
out_char_list : HEXADECIMAL |
'('HEXADECIMAL ')','repeat*'
|out_char_list ','HEXADECIMAL
;
in2out : '['i2o_list ']';
i2o_list : DECIMAL |
'('DECIMAL ')','repeat*'
i2o_list ','DECIMAL
;
out2in : '['o2i_list ']';
o2i_list : DECIMAL |
'('DECIMAL ')','repeat*'
o2i_list ','DECIMAL
;```

472   man pages section 1: User Commands • Last Revised 5 Nov 1999
property : '>[ prop_list ']'
;
prop_list : HEXADECIMAL
| '>[ HEXADECIMAL ']' 'repeat'|
| prop_list ';' HEXADECIMAL |
;
For example, the following shaping sequences can be defined:

# A simple shaping sequence:
() + 0x21 ->
{ [0x0021], [0], [0;0], [0x80] } + 0x22 ->
{ [0x0021;0x0022], [0;1], [0;0;1;1], [0x80;0x80] } + 0xc2a0 ->
{ [0x0021;0x0022;0xe030], [0;1;2], [0;0;1;1;2;2],
 [0x80;0x80;0x80;0x80] }

# A repeating shaping sequence:
() + 0x21 ->
{ ( [0x0021], [0], [0;0], [0x80] ) + 0x22 ->
{ [0x0021;0x0022], [0;1], [0;0;1;1], [0x80;0x80] } + 0xc2a2
} repeat+

The first example shows a shaping sequence such that if 0x21, 0x22, and 0xc2a0 are
the input buffer contents, it will be converted into an output buffer containing
0x0021, 0x0022, and 0xe030; an input to the output buffer containing 0, 1, and 2;
an output to the input buffer containing 0, 0, 1, 1, 2, and 2; and a property buffer
containing 0x80, 0x80, and 0x80.

The second example shows a repeating shaping sequence where, if the first input code
element is 0x21, then the second and third input code elements are 0x22 and
0xc2a2, respectively.

**EXIT STATUS**
The following exit values are returned:

0  No errors occurred and the output file was successfully created.
1  Command line options are not correctly used or unknown command line
   option specified.
2  Invalid input or output file specified.
3  The layout definitions not correctly defined.
4  No more system resource error.
6  Internal error.

**FILES**
/usr/lib/locale/common/LO_LTYPE/umle.layout.so.1
  The Universal Multiscript Layout Engine for 32-bit platforms.
/usr/lib/locale/common/LO_LTYPE/sparcv9/umle.layout.so.1
  The Universal Multiscript Layout Engine for 64-bit SPARC platform.
genlayouttbl(1)

/usr/lib/locale/common/LO_LTYPE/ia64/umle.layout.so.1
   The Universal Multiscript Layout Engine for 64-bit Intel platform.
/usr/lib/locale/locale/LO_LTYPE/layout.dat
   The binary layout table file for the locale.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWglt</td>
</tr>
</tbody>
</table>

SEE ALSO

m_create_layout(3LAYOUT), m_destroy_layout(3LAYOUT),
m_getvalues_layout(3LAYOUT), m_setvalues_layout(3LAYOUT),
m_transform_layout(3LAYOUT), m_wtransform_layout(3LAYOUT),
attributes(5), environ(5)

International Language Environments Guide

Unicode Technical Report #9: The Bidirectional Algorithm from
http://www.unicode.org/unicode/reports/
The `genmsg` utility extracts message strings with calls to `catgets(3C)` from source files and writes them in a format suitable for input to `gencat(1)`.

`genmsg` reads one or more input files and, by default, generates a message source file whose name is composed of the first input file name with `.msg`. If the `-o` option is specified, `genmsg` uses the option argument for its output file.

<table>
<thead>
<tr>
<th>Command</th>
<th>Output File</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>genmsg prog.c</code></td>
<td><code>prog.c.msg</code></td>
</tr>
<tr>
<td><code>genmsg main.c util.c tool.c</code></td>
<td><code>main.c.msg</code></td>
</tr>
<tr>
<td><code>genmsg -o prog.msg mail.c util.c</code></td>
<td><code>prog.msg</code></td>
</tr>
</tbody>
</table>

`genmsg` also allows you to invoke a preprocessor to solve the dependencies of macros and define statements for the `catgets(3C)` calls.

A project file is a database that stores a list of set numbers with their maximum message numbers. Each line in a project file is composed of a set number and its maximum message number:

```
Set_number   Maximum_message_number
```

In a project file, a line beginning with a number sign (#) or an ASCII space is considered as a comment and ignored.

`genmsg` also has the reverse operation to replace all message numbers with -1.

`genmsg` allows you to comment about messages and set numbers to inform the translator how the messages should be translated. It extracts the comment, which is surrounded with the comment indicators and has the specified tag inside the comment, from the input file and writes it with a dollar ($) prefix in the output file.

`genmsg` supports the C and C++ comment indicators, `/*`, `*/`, and `//`.

`genmsg` generates two kinds of messages for testing, prefixed messages and long messages. Prefixed messages allow you to check that your program is retrieving the messages from the message catalog. Long messages allow you to check the appearance of your window program’s initial size and position.
The following options are supported:

- **-a** Append the output into the message file `message-file` that is specified by the `-o` option. If two different messages that have the same set and message number are found, the message in the specified message file is kept and the other message in the input file is discarded.

- **-b** Place the extracted comment after the corresponding message in the output file. This option changes the placement behavior of the `-s` or `-c` option.

- **-c message-tag** Extract message comments having `message-tag` inside them from the input files and write them with a `\$` prefix as a comment in the output file.

- **-d** Include an original text of a message as a comment to be preserved along with its translations. With this option, the translator can see the original messages even after they are replaced with their translations.

- **-f** Overwrite the input files and the project file when used with the `-l` or `-r` option. With the `-r` option, `genmsg` overwrites only the input files.

- **-g project-file** Generate `project-file` that has a list of set numbers and their maximum message numbers in the input files.

- **-l project-file** Replace message numbers with the calculated numbers based upon `project-file` if the message numbers are `-l` in the input files, and then generate copies of the input files with the new message numbers and a copy of `project-file` with the new maximum message numbers. If `project-file` is not found, `genmsg` uses the maximum message number in the input file as a base number and generates `project-file`.

- **-m prefix** Fill in the message with `prefix`. This option is useful for testing.

- **-M suffix** Fill in the message with `suffix`. This option is useful for testing.

- **-n** Add comment lines to the output file indicating the file name and line number in the input files where each extracted string is encountered.

- **-o message-file** Write the output to `message-file`.

- **-p preprocessor** Invoke `preprocessor` to preprocess macros and define statements for the `catgets(3C)` calls. `genmsg` first invokes the option argument as a preprocessor and
then starts the normal process against the output from the preprocessor. `genmsg` initiates this process for all the input files.

-\texttt{-r}
  Replace message numbers with -1. This is the reverse operation of the -1 option.

-\texttt{-s set-tag}
  Extract set number comments having set-tag inside them from the input files and write them with a ‘$’ prefix as a comment in the output file. If multiple comments are specified for one set number, the first one is extracted and the rest of them are discarded.

-\texttt{-t}
  Generate a message that is three times as long as the original message. This option is useful for testing.

-\texttt{-x}
  Suppress warning messages about message and set number range checks and conflicts.

**OPERANDS**

\texttt{file} An input source file.

**EXAMPLES**

**EXAMPLE 1** Assigning message numbers and generating new files

Suppose that you have the following source and project files:

```
example% cat test.c
printf(catgets(catfd, 1, -1, "line too long");
printf(catgets(catfd, 2, -1, "invalid code");
```

The command

```
example% genmsg -l proj test.c
```

would assign the calculated message numbers based upon proj and generate the following files:

```
  test.c.msg  message file
  proj.new    updated project file
  test.c.new  new source file
```

```
example% cat test.c.msg
$quote 
$set 1
11 "line too long
$set 2
21 "invalid codeexample% cat proj.new
11 2 21example% cat test.c.new
printf(catgets(catfd, 1, 11, "line too long");
printf(catgets(catfd, 2, 21, "invalid code");
```

**EXAMPLE 2** Extracting comments into a file

The command
EXAMPLE 2 Extracting comments into a file  (Continued)

example% genmsg -s SET -c MSG test.c  example% cat test.c  /* SET: tar messages */  catgets(catfd, 1, 1, "tar: tape write error");  // MSG: don't translate "tar" and "-I".  catgets(catfd, 1, 2, "tar: missing argument for -I flag");

would extract the comments and write them in the following output file:

example% cat test.c.msg  /* SET: tar messages */  $set 1  $ /* MSG: don't translate "tar". */  1 "tar: tape write error"  $ // MSG: don't translate "tar" and "-I".  2 "tar: missing argument for -I flag"

EXAMPLE 3 Generating test messages

The command

example% genmsg -m PRE: -M :FIX test.c

would generate the following messages for testing:

example% cat test.c.msg 1 "PRE:OK:FIX" 2 "PRE:Cancel:FIX"

EXAMPLE 4 Parsing a macro and writing the extracted messages

Given the following input:

example% example.c  #include <nl_types.h>  #define MSG1 "message1"  #define MSG2 "message2"  #define MSG3 "message3"  #define MSG(n) catgets(catd, 1, n, MSG ## n)  void main(int argc, char **argv)  {  nl_catd catd = catopen(argv[0], NL_CAT_LOCALE);  (void) printf("%s0, MSG(1));  (void) printf("%s0, MSG(2));  (void) printf("%s0, MSG(3));  (void) catclose(catd);  }

The following command:

example% genmsg -p "cc -E" -o example.msg example.c

would parse the MSG macros and write the extracted messages in example.msg.

EXAMPLE 5 Assigning calculated message numbers

Suppose that you have the following header, source, and project files:
EXAMPLE 5 Assigning calculated message numbers  (Continued)

eexample% ./inc/msg.h
#define WARN_SET 1
#define ERR_SET 2
#define WARN_MSG(id, msg) catgets(catd, WARN_SET, (id), (msg))
#define ERR_MSG(id, msg) catgets(catd, ERR_SET, (id), (msg))
eexample$ example.c
#include "msg.h"
printf("%s, WARN_MSG(-1, "Warning error");
printf("%s, ERR_MSG(-1, "Fatal error");
example % proj
1 10
2 10

The command

eexample% genmsg -f -p "cc -E -I ../inc" -l proj \ 
-o example.msg example.c

would assign each of the -1 message numbers a calculated number based upon proj
and would overwrite the results to example.c and proj. Also, this command writes
the extracted messages in example.msg.

ENVIRONMENT
VARIABLES
See environ(5) for descriptions of the following environment variables that affect the
execution of genmsg: LC_MESSAGES and NLSPATH.

EXIT STATUS
The following exit values are returned:

0 Successful completion.
>0 An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWloc</td>
</tr>
</tbody>
</table>

SEE ALSO

gencat(1), catgets(3C), catopen(3C), attributes(5), environ(5)

NOTES

genmsg does not handle pointers or valuables in the catgets(3C) call. For example:

    const int set_num = 1;
    extern int msg_num(const char *);
    const char *msg = "Hello";
    catgets(catd, set_num, msg_num(msg), msg);

When the auto message numbering is turned on with a preprocessor, if there are
multiple -1’s in the catgets(3C) line, genmsg replaces all of the -1’s in the line
with a calculated number. For example, given the input:


#define MSG(id, msg) catgets(catd, 1, (id), (msg))
if (ret == -1) printf("%s, MSG(-1, "Failed")\n\nthe command

```
genmsg -l proj -p "cc -E"
```

would produce:

```
#define MSG(id, msg) catgets(catd, 1, (id), (msg))
if (ret == 1) printf("%s, MSG(1, "Failed")\n\nThe workaround would be to split it into two lines as follows:

```
if (ret == -1)
    printf("%s, MSG(-1, "Failed")\n```
NAME  
getconf – get configuration values

SYNOPSIS  
getconf [-v specification] system_var
getconf [-v specification] path_var pathname
getconf -a

DESCRIPTION  
In the first synopsis form, the getconf utility will write to the standard output the value of the variable specified by system_var, in accordance with specification if the -v option is used.

In the second synopsis form, getconf will write to the standard output the value of the variable specified by path_var for the path specified by pathname, in accordance with specification if the -v option is used.

In the third synopsis form, config will write to the standard output the names of the current system configuration variables.

The value of each configuration variable will be determined as if it were obtained by calling the function from which it is defined to be available. The value will reflect conditions in the current operating environment.

OPTIONS  
The following options are supported:

- a  
  Writes the names of the current system configuration variables to the standard output.

- v specification  
  Gives the specification which governs the selection of values for configuration variables.

OPERANDS  
The following operands are supported:

path_var  
A name of a configuration variable whose value is available from the pathconf(2) function. All of the values in the following table are supported:

<table>
<thead>
<tr>
<th>LINK_MAX</th>
<th>NAME_MAX</th>
<th>POSIX_CHOWN_RESTRICTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX_CANON</td>
<td>PATH_MAX</td>
<td>POSIX_NO_TRUNC</td>
</tr>
<tr>
<td>MAX_INPUT</td>
<td>PIPE_BUF</td>
<td>POSIX_VDISABLE</td>
</tr>
</tbody>
</table>

pathname  
A path name for which the variable specified by path_var is to be determined.

system_var  
A name of a configuration variable whose value is available from confstr(3C) or sysconf(3C). All of the values in the following table are supported:
### getconf(1)

<table>
<thead>
<tr>
<th>Environment Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARG_MAX</td>
<td>BC_BASE_MAX</td>
</tr>
<tr>
<td>BC_DIM_MAX</td>
<td>BC_SCALE_MAX</td>
</tr>
<tr>
<td>BC_STRING_MAX</td>
<td>CHAR_BIT</td>
</tr>
<tr>
<td>CHARCLASS_NAME_MAX</td>
<td>CHAR_MAX</td>
</tr>
<tr>
<td>CHAR_MIN</td>
<td>CHILD_MAX</td>
</tr>
<tr>
<td>CLK_TCK</td>
<td>COLL_WEIGHTS_MAX</td>
</tr>
<tr>
<td>CS_PATH</td>
<td>EXPR_NEST_MAX</td>
</tr>
<tr>
<td>INT_MAX</td>
<td>INT_MIN</td>
</tr>
<tr>
<td>LFS64_CFLAGS</td>
<td>LFS64_LDFLAGS</td>
</tr>
<tr>
<td>LFS64_LIBS</td>
<td>LFS64_LINTFLAGS</td>
</tr>
<tr>
<td>LFS_CFLAGS</td>
<td>LFS_LDFLAGS</td>
</tr>
<tr>
<td>LFS_LIBS</td>
<td>LFS_LINTFLAGS</td>
</tr>
<tr>
<td>LINE_MAX</td>
<td>LONG_BIT</td>
</tr>
<tr>
<td>LONG_MAX</td>
<td>LONG_MIN</td>
</tr>
<tr>
<td>MB_LEN_MAX</td>
<td>NGROUPS_MAX</td>
</tr>
<tr>
<td>NL_ARGMAX</td>
<td>NL_LANGMAX</td>
</tr>
<tr>
<td>NL_MSGMAX</td>
<td>NL_NMAX</td>
</tr>
<tr>
<td>NL_SETMAX</td>
<td>NL_TEXTMAX</td>
</tr>
<tr>
<td>NZERO</td>
<td>OPEN_MAX</td>
</tr>
<tr>
<td>POSIX2_BC_BASE_MAX</td>
<td>POSIX2_BC_DIM_MAX</td>
</tr>
<tr>
<td>POSIX2_BC_SCALE_MAX</td>
<td>POSIX2_BC_STRING_MAX</td>
</tr>
<tr>
<td>POSIX2_C_BIND</td>
<td>POSIX2_C_DEV</td>
</tr>
<tr>
<td>POSIX2_CHAR_TERM</td>
<td>POSIX2_COLL_WEIGHTS_MAX</td>
</tr>
<tr>
<td>POSIX2_C_VERSION</td>
<td>POSIX2_EXPR_NEST_MAX</td>
</tr>
<tr>
<td>POSIX2_FORT_DEV</td>
<td>POSIX2_FORT_RUN</td>
</tr>
<tr>
<td>POSIX2_LINE_MAX</td>
<td>POSIX2_LOCALEDEF</td>
</tr>
<tr>
<td>POSIX2_RE_DUP_MAX</td>
<td>POSIX2_SW_DEV</td>
</tr>
<tr>
<td>POSIX2_UPE</td>
<td>POSIX2_VERSION</td>
</tr>
<tr>
<td>__POSIX_ARG_MAX</td>
<td>__POSIX_CHILD_MAX</td>
</tr>
<tr>
<td>__POSIX_JOB_CONTROL</td>
<td>__POSIX_LINK_MAX</td>
</tr>
</tbody>
</table>

482 man pages section 1: User Commands • Last Revised 30 Jan 1998
The symbol `PATH` also is recognized, yielding the same value as the `confstr()` name value `CS_PATH`. 

User Commands 483
getconf(1)

**USAGE**

See largefile(5) for the description of the behavior of getconf when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

**EXAMPLES**

**EXAMPLE 1** Writing the value of a variable

This example illustrates the value of `{NGROUPS_MAX}`:

```bash
example$ getconf NGROUPS_MAX
```

**EXAMPLE 2** Writing the value of a variable for a specific directory

This example illustrates the value of `{NAME_MAX}` for a specific directory:

```bash
example$ getconf NAME_MAX /usr
```

**EXAMPLE 3** Dealing with unspecified results

This example shows how to deal more carefully with results that might be unspecified:

```bash
if value=$(getconf PATH_MAX /usr); then
  if [ "$value" = "undefined" ]; then
    echo PATH_MAX in /usr is infinite.
  else
    echo PATH_MAX in /usr is $value.
  fi
else
  echo Error in getconf.
fi
```

Note that

```c
sysconf(_SC_POSIX_C_BIND);
```

and

```c
system("getconf POSIX2_C_BIND");
```

in a C program could give different answers. The `sysconf` call supplies a value that corresponds to the conditions when the program was either compiled or executed, depending on the implementation; the `system` call to `getconf` always supplies a value corresponding to conditions when the program is executed.

**ENVIRONMENT VARIABLES**

See environ(5) for descriptions of the following environment variables that affect the execution of getconf: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

**EXIT STATUS**

The following exit values are returned:

- **0** The specified variable is valid and information about its current state was written successfully.
- **>0** An error occurred.

**ATTRIBUTES**

See attributes(5) for descriptions of the following attributes:
<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO  
pathconf(2), confstr(3C), sysconf(3C), attributes(5), environ(5), largefile(5)
getfacl(1)

NAME  getfacl – display discretionary file information

SYNOPSIS  getfacl [-ad] file...

DESCRIPTION  For each argument that is a regular file, special file, or named pipe, getfacl displays
the owner, the group, and the Access Control List (ACL). For each directory argument,
getfacl displays the owner, the group, and the ACL and/or the default ACL. Only
directories contain default ACLs.

getfacl may be executed on a file system that does not support ACLs. It reports the
ACL based on the base permission bits.

With no options specified, getfacl displays the filename, the file owner, the file
group owner, and both the ACL and the default ACL, if it exists.

OPTIONS  The following options are supported:

- a  Display the filename, the file owner, the file group owner, and the ACL of
the file.

- d  Display the filename, the file owner, the file group owner, and the default
ACL of the file, if it exists.

OPERANDS  The following operands are supported:

file  The path name of a regular file, special file, or named pipe.

OUTPUT  The format for ACL output is as follows:

# file: filename
# owner: uid
# group: gid
user::perm
user:uid:perm
group::perm
group:gid:perm
mask:perm
other:perm
default:user::perm
default:user:uid:perm
default:group::perm
default:group:gid:perm
default:mask:perm
default:other:perm

When multiple files are specified on the command line, a blank line separates the
ACLs for each file.

The ACL entries are displayed in the order in which they are evaluated when an
access check is performed. The default ACL entries that may exist on a directory have
no effect on access checks.
The first three lines display the filename, the file owner, and the file group owner. Note that when only the -d option is specified and the file has no default ACL, only these three lines are displayed.

The user entry without a user ID indicates the permissions that are granted to the file owner. One or more additional user entries indicate the permissions that are granted to the specified users.

The group entry without a group ID indicates the permissions that are granted to the file group owner. One or more additional group entries indicate the permissions that are granted to the specified groups.

The mask entry indicates the ACL mask permissions. These are the maximum permissions allowed to any user entries except the file owner, and to any group entries, including the file group owner. These permissions restrict the permissions specified in other entries.

The other entry indicates the permissions that are granted to others.

The default entries may exist only for directories, and indicate the default entries that are added to a file created within the directory.

The uid is a login name or a user ID if there is no entry for the uid in the system password file, /etc/passwd. The gid is a group name or a group ID if there is no entry for the gid in the system group file, /etc/group. The perm is a three character string composed of the letters representing the separate discretionary access rights: r (read), w (write), x (execute/search), or the place holder character -. The perm is displayed in the following order: rwx. If a permission is not granted by an ACL entry, the place holder character appears.

If you use the chmod(1) command to change the file group owner permissions on a file with ACL entries, both the file group owner permissions and the ACL mask are changed to the new permissions. Be aware that the new ACL mask permissions may change the effective permissions for additional users and groups who have ACL entries on the file.

In order to indicate that the ACL mask restrict an ACL entry, getfacl displays an additional tab character, pound sign ("#"), and the actual permissions granted, following the entry.

EXAMPLES

EXAMPLE 1 Displaying file information

Given file "foo", with an ACL six entries long, the command

```
host% getfacl foo
```

would print:

```
# file: foo
# owner: shea
# group: staff
```
EXAMPLE 1 Displaying file information (Continued)

user::rwx
user:spy: --
user:mookie:r --
group::r --
mask::rw --
other:: --

EXAMPLE 2 Displaying information after chmod command

Continue with the above example, after "chmod 700 foo" was issued:

```bash
host% getfacl foo
```

would print:

```bash
# file: foo
# owner: shea
# group: staff
user::rwx
user:spy: --
user:mookie:r --  #effective: --
group: --
mask: --
other: --
```

EXAMPLE 3 Displaying information when ACL contains default entries

Given directory "doo", with an ACL containing default entries, the command

```bash
host% getfacl -d doo
```

would print:

```bash
# file: doo
# owner: shea
# group: staff
default:user::rwx
default:user:spy: --
default:user:mookie:r --
default:group::r --
default:mask: --
default:other: --
```

FILES
/etc/passwd system password file
/etc/group group file

ATTRIBUTES
See attributes(5) for descriptions of the following attributes:
The output from `getfacl` is in the correct format for input to the `setfacl -f` command. If the output from `getfacl` is redirected to a file, the file may be used as input to `setfacl`. In this way, a user may easily assign one file's ACL to another file.
getfrm(1F)

NAME  getfrm – returns the current frameID number
SYNOPSIS  getfrm
DESCRIPTION  getfrm returns the current frameID number. The frameID number is a number assigned to the frame by FMLI and displayed flush left in the frame’s title bar. If a frame is closed its frameID number may be reused when a new frame is opened. getfrm takes no arguments.
EXAMPLES  EXAMPLE 1 A sample of the getfrm command.
If a menu whose frameID is 3 defines an item to have this action descriptor:
action-open text stdtext 'getfrm'
the text frame defined in the definition file stdtext would be passed the argument 3 when it is opened.
NOTES  It is not a good idea to use getfrm in a backquoted expression coded on a line by itself. Stand-alone backquoted expressions are evaluated before any descriptors are parsed, thus the frame is not yet fully current, and may not have been assigned a frameID number.
ATTRIBUTES  See attributes(5) for descriptions of the following attributes:

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO  attributes(5)
**NAME**
getitems – returns a list of currently marked menu items

**SYNOPSIS**

```bash
getitems [delimiter_string]
```

**DESCRIPTION**
The `getitems` function returns the value of `lininfo` if defined, else it returns the value of the name descriptor, for all currently marked menu items. Each value in the list is delimited by `delimiter_string`. The default value of `delimiter_string` is newline.

**EXAMPLES**

**EXAMPLE 1** A sample output of `getitems` command.

The `done` descriptor in the following menu definition file executes `getitems` when the user presses ENTER (note that the menu is multiselect):

```plaintext
Menu="Example"
multiselect=TRUE
done='getitems ":" | message'
name="Item 1"
action='message "You selected item 1"'
name="Item 2"
lininfo="This is item 2"
action='message "You selected item 2"'
name="Item 3"
action='message "You selected item 3"
```

If a user marked all three items in this menu, pressing ENTER would cause the following string to be displayed on the message line:

```
Item 1:This is item 2:Item 3
```

**NOTES**
Because `lininfo` is defined for the second menu item, its value is displayed instead of the value of the name descriptor.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

**SEE ALSO**
`attributes(5)`
getopt(1)

NAME  getopt – parse command options

SYNOPSIS  set -- 'getopt optstring $*' 

DESCRIPTION  The getopt command supersedes getopt. For more information, see NOTES below.

getopt is used to break up options in command lines for easy parsing by shell procedures and to check for legal options. optstring is a string of recognized option letters; see getopt(3C). If a letter is followed by a colon (:), the option is expected to have an argument which may or may not be separated from it by white space. The special option -- is used to delimit the end of the options. If it is used explicitly, getopt recognizes it; otherwise, getopt generates it; in either case, getopt places it at the end of the options. The positional parameters ($1 $2 ... ) of the shell are reset so that each option is preceded by a - and is in its own positional parameter; each option argument is also parsed into its own positional parameter.

EXAMPLES  EXAMPLE 1 Processing the arguments for a command

The following code fragment shows how one might process the arguments for a command that can take the options -a or -b, as well as the option -o, which requires an argument:

```
set -- getopt abo: $*
if [ $? != 0 ] then
  echo $USAGE
  exit 2
fi
for i in *
do
  case $i in
  -a | -b) FLAG=$i; shift;;
  -o) OARG=$2; shift 2;;
  --) shift; break;;
  esac
  done
```

This code accepts any of the following as equivalent:

```
cmd -aarg filename1 filename2
cmd -a -o arg filename1 filename2
cmd -oarg -a filename1 filename2
cmd -a -oarg -- filename1 filename2
```

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>
getopt(1)

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

SEE ALSO intro(1), getopts(1), getoptcvt(1), sh(1), shell_builtins(1), getopt(3C), attributes(5)

DIAGNOSTICS getopt prints an error message on the standard error when it encounters an option letter not included in optstring.

NOTES getopt will not be supported in the next major release. For this release a conversion tool has been provided, namely, getoptcvt. For more information, see getopts(1) and getoptcvt(1).

Reset optind to 1 when rescanning the options.

getopt does not support the part of Rule 8 of the command syntax standard (see intro(1)) that permits groups of option-arguments following an option to be separated by white space and quoted. For example,

cmd -a -b -o "xxx z yy" filename is not handled correctly. To correct this deficiency, use the getopts command in place of getopt.

If an option that takes an option-argument is followed by a value that is the same as one of the options listed in optstring (referring to the earlier EXAMPLES section, but using the following command line:

cmd -o -a filenamegetopt always treats it as an option-argument to -o; it never recognizes -a as an option. For this case, the for loop in the example shifts past the filename argument.
NAME
getoptcvt – convert to getopts to parse command options

SYNOPSIS
/usr/lib/getoptcvt [-b] filename
/usr/lib/getoptcvt

DESCRIPTION
/usr/lib/getoptcvt reads the shell script in filename, converts it to use getopts instead of getopt, and writes the results on the standard output.

getopts is a built-in Bourne shell command used to parse positional parameters and to check for valid options. See sh(1). It supports all applicable rules of the command syntax standard (see Rules 3-10, intro(1)). It should be used in place of the getopt command. (See the NOTES section below.) The syntax for the shell's built-in getopt command is:

getopts optstring name [argument ...]

optstring must contain the option letters the command using getopt will recognize; if a letter is followed by a colon (:) the option is expected to have an argument, or group of arguments, which must be separated from it by white space.

Each time it is invoked, getopt places the next option in the shell variable name and the index of the next argument to be processed in the shell variable OPTIND. Whenever the shell or a shell script is invoked, OPTIND is initialized to 1.

When an option requires an option-argument, getopt places it in the shell variable OPTARG.

If an illegal option is encountered, ? will be placed in name.

When the end of options is encountered, getopt exits with a non-zero exit status. The special option — may be used to delimit the end of the options.

By default, getopt parses the positional parameters. If extra arguments (argument ...) are given on the getopt command line, getopt parses them instead.

So that all new commands will adhere to the command syntax standard described in intro(1), they should use getopt or getopt to parse positional parameters and check for options that are valid for that command (see the NOTES section below).

OPTIONS
The following option is supported:

- b makes the converted script portable to earlier releases of the UNIX system. /usr/lib/getoptcvt modifies the shell script in filename so that when the resulting shell script is executed, it determines at run time whether to invoke getopt or getopt.

EXAMPLES
EXAMPLE 1 Processing the arguments for a command

The following fragment of a shell program shows how one might process the arguments for a command that can take the options -a or -b, as well as the option -o, which requires an option-argument:
EXAMPLE 1 Processing the arguments for a command (Continued)

while getopts abo: c
do
    case $c in
        a | b) FLAG=$c;;
o) OARG=$OPTARG;;
?|) echo $USAGE
            exit 2;;
esac
done
shift 'expr $OPTIND - 1'

EXAMPLE 2 Equivalent code expressions

This code accepts any of the following as equivalent:

- `cmd -a -b -o "xxx z yy" filename`
- `cmd -a -b -o "xxx z yy" -filename`
- `cmd -ab -o xxx,z,yy filename`
- `cmd -ab -o "xxx z yy" filename`
- `cmd -o xxx,z,yy b a filename`

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of getopts: LC_CTYPE, LC_MESSAGES, and NLSPATH.

OPTIND This variable is used by getoptcvt as the index of the next argument to be processed.

OPTARG This variable is used by getoptcvt to store the argument if an option is using arguments.

EXIT STATUS

The following exit values are returned:

- 0 An option, specified or unspecified by optstring, was found.
- >0 The end of options was encountered or an error occurred.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

SEE ALSO

intro(1), getopts(1), sh(1), shell_builtins(1), getopt(3C), attributes(5)

DIAGNOSTICS

getopts prints an error message on the standard error when it encounters an option letter not included in optstring.
Although the following command syntax rule (see intro(1)) relaxations are permitted under the current implementation, they should not be used because they may not be supported in future releases of the system. As in the EXAMPLES section above, -a and -b are options, and the option -o requires an option-argument. The following example violates Rule 5: options with option-arguments must not be grouped with other options:

```
example$ cmd -aboxxx filename
```

The following example violates Rule 6: there must be white space after an option that takes an option-argument:

```
example$ cmd -ab oxxx filename
```

Changing the value of the shell variable OPTIND or parsing different sets of arguments may lead to unexpected results.
The `getopts` utility can be used to retrieve options and option-arguments from a list of parameters.

Each time it is invoked, the `getopts` utility places the value of the next option in the shell variable specified by the `name` operand and the index of the next argument to be processed in the shell variable `OPTIND`. Whenever the shell is invoked, `OPTIND` will be initialised to 1.

When the option requires an option-argument, the `getopts` utility will place it in the shell variable `OPTARG`. If no option was found, or if the option that was found does not have an option-argument, `OPTARG` will be unset.

If an option character not contained in the `optstring` operand is found where an option character is expected, the shell variable specified by `name` will be set to the question-mark (`?`) character. In this case, if the first character in `optstring` is a colon (:`), the shell variable `OPTARG` will be set to the option character found, but no output will be written to standard error; otherwise, the shell variable `OPTARG` will be unset and a diagnostic message will be written to standard error. This condition is considered to be an error detected in the way arguments were presented to the invoking application, but is not an error in `getopts` processing.

If an option-argument is missing:

- If the first character of `optstring` is a colon, the shell variable specified by `name` will be set to the colon character and the shell variable `OPTARG` will be set to the option character found.
- Otherwise, the shell variable specified by `name` will be set to the question-mark character (`?`), the shell variable `OPTARG` will be unset, and a diagnostic message will be written to standard error. This condition is considered to be an error detected in the way arguments were presented to the invoking application, but is not an error in `getopts` processing; a diagnostic message will be written as stated, but the exit status will be zero.

When the end of options is encountered, the `getopts` utility will exit with a return value greater than zero; the shell variable `OPTIND` will be set to the index of the first non-option-argument, where the first `--` argument is considered to be an option-argument if there are no other non-option-arguments appearing before it, or the value $# + 1 if there are no non-option-arguments; the `name` variable will be set to the question-mark character. Any of the following identifies the end of options: the special option `--`, finding an argument that does not begin with a `--`, or encountering an error.
The shell variables $OPTIND$ and $OPTARG$ are local to the caller of `getopts` and are not exported by default.

The shell variable specified by the `name` operand, $OPTIND$ and $OPTARG$ affect the current shell execution environment.

If the application sets $OPTIND$ to the value 1, a new set of parameters can be used: either the current positional parameters or new `arg` values. Any other attempt to invoke `getopts` multiple times in a single shell execution environment with parameters (positional parameters or `arg` operands) that are not the same in all invocations, or with an $OPTIND$ value modified to be a value other than 1, produces unspecified results.

`getopts` is a built-in Bourne shell command used to parse positional parameters and to check for valid options. See `sh(1)`. It supports all applicable rules of the command syntax standard (see Rules 3-10, `intro(1)`). It should be used in place of the `getopt` command.

`optstring` must contain the option letters the command using `getopts` will recognize; if a letter is followed by a colon, the option is expected to have an argument, or group of arguments, which must be separated from it by white space.

Each time it is invoked, `getopts` places the next option in the shell variable `name` and the index of the next argument to be processed in the shell variable $OPTIND$. Whenever the shell or a shell script is invoked, $OPTIND$ is initialized to 1.

When an option requires an option-argument, `getopts` places it in the shell variable $OPTARG$.

If an illegal option is encountered, `?` will be placed in `name`.

When the end of options is encountered, `getopts` exits with a non-zero exit status. The special option `-` may be used to delimit the end of the options.

By default, `getopts` parses the positional parameters. If extra arguments (`argument ...`) are given on the `getopts` command line, `getopts` parses them instead.

`/usr/lib/getoptcvt` reads the shell script in `filename`, converts it to use `getopts` instead of `getopt`, and writes the results on the standard output.

So that all new commands will adhere to the command syntax standard described in `intro(1)`, they should use `getopts` or `getopt` to parse positional parameters and check for options that are valid for that command.

`getopts` prints an error message on the standard error when it encounters an option letter not included in `optstring`.

Although the following command syntax rule (see `intro(1)`) relaxations are permitted under the current implementation, they should not be used because they may not be supported in future releases of the system. As in the EXAMPLES section below, `-a` and `-b` are options, and the option `-c` requires an option-argument.
The following example violates Rule 5: options with option-arguments must not be grouped with other options:

```
example cmd -aboxxx filename
```

The following example violates Rule 6: there must be white space after an option that takes an option-argument:

```
example cmd -ab oxxx filename
```

Changing the value of the shell variable `OPTIND` or parsing different sets of arguments may lead to unexpected results.

**ksh**  Checks `arg` for legal options. If `arg` is omitted, the positional parameters are used. An option argument begins with a `+` or a `−`. An option not beginning with `+` or `-` or the argument `−` ends the options. `optstring` contains the letters that `getopts` recognizes. If a letter is followed by a `:`, that option is expected to have an argument. The options can be separated from the argument by blanks.

`getopts` places the next option letter it finds inside variable `name` each time it is invoked with a `+` prepended when `arg` begins with a `+`. The index of the next `arg` is stored in `OPTIND`. The option argument, if any, gets stored in `OPTARG`.

A leading `:` in `optstring` causes `getopts` to store the letter of an invalid option in `OPTARG`, and to set `name` to `?` for an unknown option and to `:` when a required option is missing. Otherwise, `getopts` prints an error message. The exit status is non-zero when there are no more options.

For a further discussion of the Korn shell’s `getopts` built-in command, see the previous discussion in the Bourne shell (`sh`) section of this manpage.

**OPERANDS**  The following operands are supported:

- `optstring`  A string containing the option characters recognised by the utility invoking `getopts`. If a character is followed by a colon, the option will be expected to have an argument, which should be supplied as a separate argument. Applications should specify an option character and its option-argument as separate arguments, but `getopts` will interpret the characters following an option character requiring arguments as an argument whether or not this is done. An explicit null option-argument need not be recognised if it is not supplied as a separate argument when `getopts` is invoked; see `getopt(3C)`. The characters question-mark (`?`) and colon (`:`) must not be used as option characters by an application. The use of other option characters that are not alphanumeric produces unspecified results. If the option-argument is not supplied as a separate argument from the option character, the value in `OPTARG` will be stripped of the option character and the −.
getopts(1)

The first character in *optstring* will determine how *getopts* will behave if an option character is not known or an option-argument is missing.

**name**

The name of a shell variable that will be set by the *getopts* utility to the option character that was found.

The *getopts* utility by default will parse positional parameters passed to the invoking shell procedure. If *args* are given, they will be parsed instead of the positional parameters.

**USAGE**

Since *getopts* affects the current shell execution environment, it is generally provided as a shell regular built-in. If it is called in a subshell or separate utility execution environment, such as one of the following:

```
(getopts abc value "$@")
nosh getopts ...
find . -exec getopts ... \\
```

it will not affect the shell variables in the caller’s environment.

Notice that shell functions share *OPTIND* with the calling shell even though the positional parameters are changed. Functions that want to use *getopts* to parse their arguments will usually want to save the value of *OPTIND* on entry and restore it before returning. However, there will be cases when a function will want to change *OPTIND* for the calling shell.

**EXAMPLES**

**EXAMPLE 1** Parsing and displaying arguments

The following example script parses and displays its arguments:

```bash
aflag=
bflag=
while getopts ab: name do
case $name in
  a) aflag=1;;
b) bflag=1
     bval="$OPTARG";;
?) printf "Usage: %s: [-a] [-b value] args\n" $0
     exit 2;;
esac
done
if [ ! -z "$aflag" ]; then
  printf "Option -a specified\n"
fi
if [ ! -z "$bflag" ]; then
  printf 'Option -b "$s" specified\n' "$bval"
fi
shift $((OPTIND - 1))
printf 'Remaining arguments are: %s\n' "$*"
```

getopts(1)
EXAMPLE 2  Processing arguments for a command with options

The following fragment of a shell program shows how one might process the arguments for a command that can take the options \(-a\) or \(-b\), as well as the option \(-o\), which requires an option-argument:

```bash
while getopts abo: c
  do
    case $c in
      a | b) FLAG=$c;;
      o) OARG=$OPTARG;;
      \?) echo $USAGE
        exit 2;;
      esac
    done
    shift \((\$(expr $OPTIND \(-\) 1))\)
  done
```

EXAMPLE 3  Equivalent code expressions

This code accepts any of the following as equivalent:

- `cmd -a -b -o "xxx z yy" filename`
- `cmd -a -b -o "xxx z yy" -- filename`
- `cmd -ab -o xxx,z,yy filename`
- `cmd -ab -o "xxx z yy" filename`
- `cmd -o xxx,z,yy -b -a filename`

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of `getopts`: LC_CTYPE, LC_MESSAGES, and NLSPATH.

- **OPTIND**
  This variable is used by `getopts` as the index of the next argument to be processed.

- **OPTARG**
  This variable is used by `getopts` to store the argument if an option is using arguments.

EXIT STATUS

The following exit values are returned:

- **0**  An option, specified or unspecified by `optstring`, was found.
- **>0**  The end of options was encountered or an error occurred.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO intro(1), getoptcvt(1), ksh(1), sh(1), getopt(3C), attributes(5), environ(5)
Whenever an error is detected and the first character in the `optstring` operand is not a colon (:`), a diagnostic message will be written to standard error with the following information in an unspecified format:

- The invoking program name will be identified in the message. The invoking program name will be the value of the shell special parameter `0` at the time the `getopts` utility is invoked. A name equivalent to `basename "$0"` may be used.

- If an option is found that was not specified in `optstring`, this error will be identified and the invalid option character will be identified in the message.

- If an option requiring an option-argument is found, but an option-argument is not found, this error will be identified and the invalid option character will be identified in the message.
The gettext utility retrieves a translated text string corresponding to string `msgid` from a message object generated with `msgfmt(1)`. The message object name is derived from the optional argument `textdomain` if present, otherwise from the `TEXTDOMAIN` environment. If no domain is specified, or if a corresponding string cannot be found, `gettext` prints `msgid`.

Ordinarily, `gettext` looks for its message object in `/usr/lib/locale/lang/LC_MESSAGES` where `lang` is the locale name. If present, the `TEXTDOMAINDIR` environment variable replaces the pathname component up to `lang`.

This command interprets C escape sequences such as `\t` for tab. Use `\\` to print a backslash. To produce a message on a line of its own, either enter `\n` at the end of `msgid`, or use this command in conjunction with `printf(1)`.

When used with the `-s` option, `gettext` behaves like `echo(1)`. But it does not simply copy its arguments to standard output. Instead, those messages found in the selected catalog are translated.

The following options are supported:

- `-d textdomain`  Retrieves translated messages from the domain `textdomain`, if `textdomain` is not specified as an operand.
- `--domain=textdomain`  Same as `-d`.
- `-e`  Enables expansion of some escape sequences if used with the `-s` option.
- `-n`  Suppresses trailing newline if used with the `-s` option.
- `-s`  Behaves like `echo(1)` (see DESCRIPTION above). If the `-s` option is specified, no expansion of C escape sequences is performed and a newline character is appended to the output, by default.

The following operands are supported:

- `textdomain`  A domain name used to retrieve the messages. This overrides the specification by the `-d` or `--domain` options, if present.
- `msgid`  A key to retrieve the localized message.

The following environment variables are supported:

- `LANG`  Specifies locale name.
- `LC_MESSAGES`  Specifies messaging locale, and if present overrides `LANG` for messages.
gettext(1)

TEXTDOMAIN Specifies the text domain name, which is identical to the message object filename without .mo suffix.

TEXTDOMAINIR Specifies the pathname to the message database. If present, replaces /usr/lib/locale.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO echo(1), msgfmt(1), printf(1), gettext(3C), setlocale(3C), attributes(5)

NOTES This is the shell equivalent of the library routine gettext(3C).
gettxt(1)

NAME
gettxt – retrieve a text string from a message database

SYNOPSIS
gettxt msgfile : msgnum [dflt_msg]

DESCRIPTION
gettxt retrieves a text string from a message file in the directory
/usr/lib/locale/locale/LC_MESSAGES. The directory name locale corresponds to
the language in which the text strings are written; see setlocale(3C).

msgfile
Name of the file in the directory
/usr/lib/locale/locale/LC_MESSAGES to retrieve msgnum
from. The name of msgfile can be up to 14 characters in length, but
may not contain either \0 (null) or the ASCII code for / (slash) or
: (colon).

msgnum
Sequence number of the string to retrieve from msgfile. The strings
in msgfile are numbered sequentially from 1 to n, where n is the
number of strings in the file.

dflt_msg
Default string to be displayed if gettxt fails to retrieve msgnum
from msgfile. Nongraphic characters must be represented as
alphabetic escape sequences.

The text string to be retrieved is in the file msgfile, created by the mkmsgs(1) utility and
installed under the directory /usr/lib/locale/locale/LC_MESSAGES. You control
which directory is searched by setting the environment variable LC_MESSAGES. If
LC_MESSAGES is not set, the environment variable LANG will be used. If LANG is not
set, the files containing the strings are under the directory
/usr/lib/locale/C/LC_MESSAGES.

If gettxt fails to retrieve a message in the requested language, it will try to retrieve
the same message from /usr/lib/locale/C/LC_MESSAGES/ msgfile. If this also
fails, and if dflt_msg is present and non-null, then it will display the value of dflt_msg;
if dflt_msg is not present or is null, then it will display the string Message not
found!.

EXAMPLES
EXAMPLE 1 The environment variables LANG and LC_MESSAGES.

If the environment variables LANG or LC_MESSAGES have not been set to other than
their default values, the following example:

eexample$ gettxt UX:10 "hello world\n"

will try to retrieve the 10th message from /usr/lib/locale/C/UX/msgfile. If the
retrieval fails, the message "hello world," followed by a newline, will be displayed.

ENVIRONMENT
VARIABLES
See environ(5) for descriptions of the following environment variables that affect the
execution of gettxt: LC_CTYPE and LC_MESSAGES.

LC_CTYPE
Determines how gettxt handles characters. When
LC_CTYPE is set to a valid value, gettxt can display
and handle text and filenames containing valid
gettxt(1)

characters for that locale. gettext can display and handle Extended Unix Code (EUC) characters where any individual character can be 1, 2, or 3 bytes wide. gettext can also handle EUC characters of 1, 2, or more column widths. In the "C" locale, only characters from ISO 8859-1 are valid.

LC_MESSAGES

Determines how diagnostic and informative messages are presented. This includes the language and style of the messages, and the correct form of affirmative and negative responses. In the "C" locale, the messages are presented in the default form found in the program itself (in most cases, U.S. English).

FILES

/usr/lib/locale/C/LC_MESSAGES/*
    default message files created by mkmsgs(1)

/usr/lib/locale/locale/LC_MESSAGES/*
    message files for different languages created by mkmsgs(1)

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWloc</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

SEE ALSO

exstr(1), mkmsgs(1), srchtxt(1), gettext(3C), setlocale(3C), attributes(5), environ(5)
NAME
glob – shell built-in function to expand a word list
csh  glob  wordlist
csh  glob performs filename expansion on  wordlist.  Like  echo(1),  but  no  ‘\’  escapes  are  
recognized. Words are delimited by null characters in the output.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO  csh(1), echo(1), attributes(5)
gprof(1)

NAME   gprof – display call-graph profile data

SYNOPSIS gprof [-abcDlzs] [-e function-name] [-E function-name] [-f function-name] [-F function-name] [image-file [profile-file...]] [-n number of functions]

DESCRIPTION The gprof utility produces an execution profile of a program. The effect of called routines is incorporated in the profile of each caller. The profile data is taken from the call graph profile file that is created by programs compiled with the -xpg option of cc(1), or by the -pg option with other compilers, or by setting the LD_PROFILE environment variable for shared objects. See ld.so.1(1). These compiler options also link in versions of the library routines which are compiled for profiling. The symbol table in the executable image file image-file (a.out by default) is read and correlated with the call graph profile file profile-file (gmon.out by default).

First, execution times for each routine are propagated along the edges of the call graph. Cycles are discovered, and calls into a cycle are made to share the time of the cycle. The first listing shows the functions sorted according to the time they represent, including the time of their call graph descendants. Below each function entry is shown its (direct) call-graph children and how their times are propagated to this function. A similar display above the function shows how this function’s time and the time of its descendants are propagated to its (direct) call-graph parents.

Cycles are also shown, with an entry for the cycle as a whole and a listing of the members of the cycle and their contributions to the time and call counts of the cycle.

Next, a flat profile is given, similar to that provided by prof(1). This listing gives the total execution times and call counts for each of the functions in the program, sorted by decreasing time. Finally, an index is given, which shows the correspondence between function names and call-graph profile index numbers.

A single function may be split into subfunctions for profiling by means of the MARK macro. See prof(5).

Beware of quantization errors. The granularity of the sampling is shown, but remains statistical at best. It is assumed that the time for each execution of a function can be expressed by the total time for the function divided by the number of times the function is called. Thus the time propagated along the call-graph arcs to parents of that function is directly proportional to the number of times that arc is traversed.

The profiled program must call exit(2) or return normally for the profiling information to be saved in the gmon.out file.

OPTIONS The following options are supported:

- a Suppress printing statically declared functions. If this option is given, all relevant information about the static function (for instance, time samples, calls to other functions, calls from other functions) belongs to the function loaded just before the static function in the a.out file.
Brief. Suppress descriptions of each field in the profile.

Discover the static call-graph of the program by a heuristic which examines the text space of the object file. Static-only parents or children are indicated with call counts of 0. Note that for dynamically linked executables, the linked shared objects’ text segments are not examined.

Demangle C++ symbol names before printing them out.

Produce a profile file gmon. sum that represents the difference of the profile information in all specified profile files. This summary profile file may be given to subsequent executions of gprof (also with -D) to summarize profile data across several runs of an a.out file. See also the -s option.

As an example, suppose function A calls function B n times in profile file gmon. sum, and m times in profile file gmon. out. With -D, a new gmon. sum file will be created showing the number of calls from A to B as n - m.

Suppress printing the graph profile entry for routine function-name and all its descendants (unless they have other ancestors that are not suppressed). More than one -e option may be given. Only one function-name may be given with each -e option.

Suppress printing the graph profile entry for routine function-name (and its descendants) as -e, below, and also exclude the time spent in function-name (and its descendants) from the total and percentage time computations. More than one -E option may be given. For example:

' -E mcount -E mcleanup'

is the default.

Print the graph profile entry only for routine function-name and its descendants. More than one -f option may be given. Only one function-name may be given with each -f option.

Print the graph profile entry only for routine function-name and its descendants (as -f, below) and also use only the times of the printed routines in total
time and percentage computations. More than one -F option may be given. Only one function-name may be given with each -F option. The -F option overrides the -E option.

-1

Suppress the reporting of graph profile entries for all local symbols. This option would be the equivalent of placing all of the local symbols for the specified executable image on the -E exclusion list.

-n

Limits the size of flat and graph profile listings to the top n offending functions.

-s

Produce a profile file gmon.sum which represents the sum of the profile information in all of the specified profile files. This summary profile file may be given to subsequent executions of gprof (also with -s) to accumulate profile data across several runs of an a.out file. See also the -D option.

-z

Display routines which have zero usage (as indicated by call counts and accumulated time). This is useful in conjunction with the -c option for discovering which routines were never called. Note that this has restricted use for dynamically linked executables, since shared object text space will not be examined by the -c option.

ENVIRONMENT VARIABLES

PROFDIR

If this environment variable contains a value, place profiling output within that directory, in a file named pid.programname. pid is the process ID and programname is the name of the program being profiled, as determined by removing any path prefix from the argv[0] with which the program was called. If the variable contains a null value, no profiling output is produced. Otherwise, profiling output is placed in the file gmon.out.

FILES

a.out	executable file containing namelist
gmon.out
dynamic call-graph and profile
gmon.sum
summarized dynamic call-graph and profile

$PROFDIR/pid.programname

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWbtool</td>
</tr>
</tbody>
</table>
See Also
cc(1), ld.so.1(1), prof(1), exit(2), pcsample(2), profil(2), malloc(3C), malloc(3MALLOCS), monitor(3C), attributes(5), prof(5)


Linker and Libraries Guide

If the executable image has been stripped and has no symbol table (.symtab), then gprof will read the dynamic symbol table (.dyntab), if present. If the dynamic symbol table is used, then only the information for the global symbols will be available, and the behavior will be identical to the -a option.

LD_LIBRARY_PATH must not contain /usr/lib as a component when compiling a program for profiling. If LD_LIBRARY_PATH contains /usr/lib, the program will not be linked correctly with the profiling versions of the system libraries in /usr/lib/libp.

The times reported in successive identical runs may show variances because of varying cache-hit ratios that result from sharing the cache with other processes. Even if a program seems to be the only one using the machine, hidden background or asynchronous processes may blur the data. In rare cases, the clock ticks initiating recording of the program counter may 'beat' with loops in a program, grossly distorting measurements. Call counts are always recorded precisely, however.

Only programs that call exit or return from main are guaranteed to produce a profile file, unless a final call to monitor is explicitly coded.

Functions such as mcount(), _mcount(), moncontrol(), _moncontrol(), monitor(), and _monitor() may appear in the gprof report. These functions are part of the profiling implementation and thus account for some amount of the runtime overhead. Since these functions are not present in an unprofiled application, time accumulated and call counts for these functions may be ignored when evaluating the performance of an application.

64-bit profiling
64-bit profiling may be used freely with dynamically linked executables, and profiling information is collected for the shared objects if the objects are compiled for profiling. Care must be applied to interpret the profile output, since it is possible for symbols from different shared objects to have the same name. If name duplication occurs in the profile output, the module id prefix before the symbol name in the symbol index listing can be used to identify the appropriate module for the symbol.

When using the -s or -D option to sum multiple profile files, care must be taken not to mix 32-bit profile files with 64-bit profile files.

32-bit profiling
32-bit profiling may be used with dynamically linked executables, but care must be applied. In 32-bit profiling, shared objects cannot be profiled with gprof. Thus, when a profiled, dynamically linked program is executed, only the 'main' portion of the
image is sampled. This means that all time spent outside of the "main" object, that is, time spent in a shared object, will not be included in the profile summary; the total time reported for the program may be less than the total time used by the program.

Because the time spent in a shared object cannot be accounted for, the use of shared objects should be minimized whenever a program is profiled with gprof. If desired, the program should be linked to the profiled version of a library (or to the standard archive version if no profiling version is available), instead of the shared object to get profile information on the functions of a library. Versions of profiled libraries may be supplied with the system in the /usr/lib/libp directory. Refer to compiler driver documentation on profiling.

Consider an extreme case. A profiled program dynamically linked with the shared C library spends 100 units of time in some libc routine, say, malloc(). Suppose malloc() is called only from routine B and B consumes only 1 unit of time. Suppose further that routine A consumes 10 units of time, more than any other routine in the "main" (profiled) portion of the image. In this case, gprof will conclude that most of the time is being spent in A and almost no time is being spent in B. From this it will be almost impossible to tell that the greatest improvement can be made by looking at routine B and not routine A. The value of the profiler in this case is severely degraded; the solution is to use archives as much as possible for profiling.

**BUGS**

Parents which are not themselves profiled will have the time of their profiled children propagated to them, but they will appear to be spontaneously invoked in the call-graph listing, and will not have their time propagated further. Similarly, signal catchers, even though profiled, will appear to be spontaneous (although for more obscure reasons). Any profiled children of signal catchers should have their times propagated properly, unless the signal catcher was invoked during the execution of the profiling routine, in which case all is lost.
NAME
graph – draw a graph

SYNOPSIS
graph [-d spacing [start]] [-b] [-c string] [-g gridstyle] [-l label]
[-m connectmode] [-s] [-x [l] lower [upper spacing]] [-y [l] lower
[upper spacing]] [-h fraction] [-w fraction] [-r fraction] [-u fraction]
[-t] ...

DESCRIPTION
graph with no options takes pairs of numbers from the standard input as abscissaes
and ordinates of a graph. Successive points are connected by straight lines. The
standard output from graph contains plotting instructions suitable for input to
plot(1B) or to the command lpr -g (see lpr(1B)).

If the coordinates of a point are followed by a nonnumeric string, that string is printed
as a label beginning on the point. Labels may be surrounded with quotes "...",
in which case they may be empty or contain blanks and numbers; labels never contain
NEWLINE characters.

A legend indicating grid range is produced with a grid unless the -s option is present.

OPTIONS
Each option is recognized as a separate argument. If a specified lower limit exceeds
the upper limit, the axis is reversed.

-a spacing [start]
Supply abscissaes automatically (they are missing from the input); spacing is the
spacing (default 1). start is the starting point for automatic abscissaes (default 0 or
lower limit given by -x).

-b
Break (disconnect) the graph after each label in the input.

-c string
String is the default label for each point.

-g gridstyle
Gridstyle is the grid style: 0 no grid, 1 frame with ticks, 2 full grid (default).

-l label
label is label for graph.

-m connectmode
Mode (style) of connecting lines: 0 disconnected, 1 connected (default). Some
devices give distinguishable line styles for other small integers.

-s
Save screen, do not erase before plotting.

-x [l] lower [upper [spacing]]
If l is present, x axis is logarithmic. lower and upper are lower (and upper) x limits.
spacing, if present, is grid spacing on x axis. Normally these quantities are
determined automatically.
-y [ l ] lower [ upper [ spacing ] ]

If l is present, y axis is logarithmic. lower and upper are lower (and upper) y limits. spacing, if present, is grid spacing on y axis. Normally these quantities are determined automatically.

-h fraction
fraction of space for height.

-w fraction
fraction of space for width.

-x fraction
fraction of space to move right before plotting.

-u fraction
fraction of space to move up before plotting.

-t
Transpose horizontal and vertical axes. Option -x now applies to the vertical axis.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
</tbody>
</table>

SEE ALSO
lpr(1B), plot(1B), spline(1), plot(3PLOT), attributes(5)

BUGS
graph stores all points internally and drops those for which there is no room.
Segments that run out of bounds are dropped, not windowed.
Logarithmic axes may not be reversed.
grep utility searches text files for a pattern and prints all lines that contain that pattern. It uses a compact non-deterministic algorithm.

Be careful using the characters $, *, [, ^, |, (, )], and \ in the pattern_list because they are also meaningful to the shell. It is safest to enclose the entire pattern_list in single quotes ‘...’.

If no files are specified, grep assumes standard input. Normally, each line found is copied to standard output. The file name is printed before each line found if there is more than one input file.

The /usr/bin/grep utility uses limited regular expressions like those described on the regexp(5) manual page to match the patterns.

The options -E and -F affect the way /usr/xpg4/bin/grep interprets pattern_list. If -E is specified, /usr/xpg4/bin/grep interprets pattern_list as a full regular expression (see -E for description). If -F is specified, grep interprets pattern_list as a fixed string. If neither are specified, grep interprets pattern_list as a basic regular expression as described on regexp(5) manual page.

The following options are supported for both /usr/bin/grep and /usr/xpg4/bin/grep:

- **-b** Precede each line by the block number on which it was found. This can be useful in locating block numbers by context (first block is 0).
- **-c** Print only a count of the lines that contain the pattern.
- **-h** Prevents the name of the file containing the matching line from being appended to that line. Used when searching multiple files.
- **-i** Ignore upper/lower case distinction during comparisons.
- **-l** Print only the names of files with matching lines, separated by NEWLINE characters. Does not repeat the names of files when the pattern is found more than once.
- **-n** Precede each line by its line number in the file (first line is 1).
- **-s** Suppress error messages about nonexistent or unreadable files.
grep(1)

- v  Print all lines except those that contain the pattern.
- w  Search for the expression as a word as if surrounded by \< and \>.

The following options are supported for /usr/xpg4/bin/grep only:

- e pattern_list  Specify one or more patterns to be used during the search for input. Patterns in pattern_list must be separated by a NEWLINE character. A null pattern can be specified by two adjacent newline characters in pattern_list. Unless the -E or -F option is also specified, each pattern will be treated as a basic regular expression. Multiple -e and -E options are accepted by grep. All of the specified patterns are used when matching lines, but the order of evaluation is unspecified.

- E  Match using full regular expressions. Treat each pattern specified as a full regular expression. If any entire full regular expression pattern matches an input line, the line will be matched. A null full regular expression matches every line. Each pattern will be interpreted as a full regular expression as described on the regex(5) manual page, except for \ (and \), and including:

1. A full regular expression followed by + that matches one or more occurrences of the full regular expression.
2. A full regular expression followed by ? that matches 0 or 1 occurrences of the full regular expression.
3. Full regular expressions separated by | or by a new-line that match strings that are matched by any of the expressions.
4. A full regular expression that may be enclosed in parentheses () for grouping.

The order of precedence of operators is [ ], then * ?, +, then concatenation, then | and new-line.

- F pattern_file  Read one or more patterns from the file named by the path name pattern_file. Patterns in pattern_file are terminated by a NEWLINE character. A null pattern can be specified by an empty line in pattern_file. Unless the -E or -F option is also specified, each pattern will be treated as a basic regular expression.

- F  Match using fixed strings. Treat each pattern specified as a string instead of a regular expression. If an input line contains any of the patterns as a contiguous sequence of bytes, the line will be matched. A null string matches every line. See fgrep(1) for more information.

- q  Quiet. Do not write anything to the standard output, regardless of matching lines. Exit with zero status if an input line is selected.

- x  Consider only input lines that use all characters in the line to match an entire fixed string or regular expression to be matching
The following operands are supported:

file
A path name of a file to be searched for the patterns. If no file operands are specified, the standard input will be used.

pattern
Specify a pattern to be used during the search for input.

The -e pattern_list option has the same effect as the pattern_list operand, but is useful when pattern_list begins with the hyphen delimiter. It is also useful when it is more convenient to provide multiple patterns as separate arguments.

Multiple -e and -f options are accepted and grep will use all of the patterns it is given while matching input text lines. (Note that the order of evaluation is not specified. If an implementation finds a null string as a pattern, it is allowed to use that pattern first, matching every line, and effectively ignore any other patterns.)

The -q option provides a means of easily determining whether or not a pattern (or string) exists in a group of files. When searching several files, it provides a performance improvement (because it can quit as soon as it finds the first match) and requires less care by the user in choosing the set of files to supply as arguments (because it will exit zero if it finds a match even if grep detected an access or read error on earlier file operands).

See largefile(5) for the description of the behavior of grep when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXAMPLE 1 Finding all uses of a word

To find all uses of the word “Posix” (in any case) in the file text.mm, and write with line numbers:

```
example$ /usr/bin/grep -i -n posix text.mm
```

EXAMPLE 2 Finding all empty lines

To find all empty lines in the standard input:

```
example$ /usr/bin/grep ^$ 
```

or

```
example$ /usr/bin/grep -v .
```
EXAMPLE 2 Finding all empty lines (Continued)

EXAMPLE 3 Finding lines containing strings

Both of the following commands print all lines containing strings abc or def or both:

directory$ /usr/xpg4/bin/grep -E 'abc def'
directory$ /usr/xpg4/bin/grep -F 'abc def'

EXAMPLE 4 Finding lines with matching strings

Both of the following commands print all lines matching exactly abc or def:

directory$ /usr/xpg4/bin/grep -E '^abc$ ^def$'
directory$ /usr/xpg4/bin/grep -F -x 'abc def'

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of grep: LC_COLLATE, LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS

The following exit values are returned:

0 One or more matches were found.
1 No matches were found.
2 Syntax errors or inaccessible files (even if matches were found).

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWxcu4</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

SEE ALSO


_LINES ARE LIMITED ONLY BY THE SIZE OF THE AVAILABLE VIRTUAL MEMORY. IF THERE IS A LINE WITH EMBEDDED NULLS, GREP WILL ONLY MATCH UP TO THE FIRST NULL; IF IT MATCHES, IT WILL PRINT THE ENTIRE LINE.

518 man pages section 1: User Commands • Last Revised 20 Oct 1997
The results are unspecified if input files contain lines longer than LINE_MAX bytes or contain binary data. LINE_MAX is defined in /usr/include/limits.h.
groups(1)

NAME    groups – print group membership of user

SYNOPSIS groups [user...]

DESCRIPTION The command groups prints on standard output the groups to which you or the
optionally specified user belong. Each user belongs to a group specified in
/etc/passwd and possibly to other groups as specified in /etc/group. Note that
/etc/passwd specifies the numerical ID (gid) of the group. The groups command
converts gid to the group name in the output.

EXAMPLES The output takes the following form:

example$ groups tester01 tester02
  tester01 : staff
  tester02 : staff
example$

FILES /etc/passwd
       /etc/group

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO group(4), passwd(4), attributes(5)
NAME | groups – display a user’s group memberships

SYNOPSIS | /usr/ucb/groups [user...]

DESCRIPTION | With no arguments, groups displays the groups to which you belong; else it displays the groups to which the user belongs. Each user belongs to a group specified in the password file /etc/passwd and possibly to other groups as specified in the file /etc/group. If you do not own a file but belong to the group which it is owned by then you are granted group access to the file.

FILES | /etc/passwd
        /etc/group

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

SEE ALSO | getgroups(2), attributes(5)

NOTES | This command is obsolete.
grpck(1B)

NAME    grpck – check group database entries

SYNOPSIS  /etc/grpck [filename]

DESCRIPTION  The grpck utility checks that a file in group(4) does not contain any errors; it checks the /etc/group file by default.

FILES  /etc/group

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO  groups(1), group(4), passwd(4), attributes(5)

DIAGNOSTICS  Too many/few fields

  An entry in the group file does not have the proper number of fields.

No group name

  The group name field of an entry is empty.

Bad character(s) in group name

  The group name in an entry contains characters other than lower-case letters and digits.

Invalid GID

  The group ID field in an entry is not numeric or is greater than 65535.

Null login name

  A login name in the list of login names in an entry is null.

Logname not found in password file

  A login name in the list of login names in an entry is not in the password file.

Line too long

  A line (including the newline character) in the group file exceeds the maximum length of 512 characters.

Duplicate logname entry

  A login name appears more than once in the list of login names for a group file entry.

Out of memory

  The program cannot allocate memory in order to continue.

Maximum groups exceeded for logname

  A login name’s group membership exceeds the maximum, NGROUPS_MAX.
hash, rehash, unhash, hashstat – evaluate the internal hash table of the contents of directories

SYNOPSIS

```
/usr/bin/hash [utility]
```

```
/usr/bin/hash [-r]
```

```
sh hash [-r] [name...]
```

```
csh rehash
```

```
unhash
```

```
hashstat
```

```
ksh hash [name...]
```

`/usr/bin/hash` The `/usr/bin/hash` utility affects the way the current shell environment remembers the locations of utilities found. Depending on the arguments specified, it adds utility locations to its list of remembered locations or it purges the contents of the list. When no arguments are specified, it reports on the contents of the list.

Utilities provided as built-ins to the shell are not reported by `hash`.

`sh` For each `name`, the location in the search path of the command specified by `name` is determined and remembered by the shell. The `-r` option to the `hash` built-in causes the shell to forget all remembered locations. If no arguments are given, `hash` provides information about remembered commands. The `Hits` column of output is the number of times a command has been invoked by the shell process. The `Cost` column of output is a measure of the work required to locate a command in the search path. If a command is found in a "relative" directory in the search path, after changing to that directory, the stored location of that command is recalculated. Commands for which this will be done are indicated by an asterisk (*) adjacent to the `Hits` information. `Cost` will be incremented when the recalculation is done.

`csh` rehash recomputes the internal hash table of the contents of directories listed in the `path` environmental variable to account for new commands added.

unhash disables the internal hash table.

hashstat prints a statistics line indicating how effective the internal hash table has been at locating commands (and avoiding execs). An exec is attempted for each component of the `path` where the hash function indicates a possible hit and in each component that does not begin with a ‘/’.

`ksh` For each `name`, the location in the search path of the command specified by `name` is determined and remembered by the shell. If no arguments are given, `hash` provides information about remembered commands.

OPERANDS

```
The following operand is supported by `hash`:
```

```
utility The name of a utility to be searched for and added to the list of remembered locations.
```
The standard output of `hash` is used when no arguments are specified. Its format is unspecified, but includes the pathname of each utility in the list of remembered locations for the current shell environment. This list consists of those utilities named in previous `hash` invocations that have been invoked, and may contain those invoked and found through the normal command search process.

See `environ(5)` for descriptions of the following environment variables that affect the execution of `hash`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

`PATH` Determine the location of utility.

The following exit values are returned by `hash`:

- 0 Successful completion.
- >0 An error occurred.

See also `csh(1), ksh(1), sh(1), attributes(5), environ(5)`

<table>
<thead>
<tr>
<th>TYPE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>
head – display first few lines of files

head [-number | -n number] [filename...]

The head utility copies the first number of lines of each filename to the standard output. If no filename is given, head copies lines from the standard input. The default value of number is 10 lines.

When more than one file is specified, the start of each file will look like:

== filename ==

Thus, a common way to display a set of short files, identifying each one, is:

example% head -9999 filename1 filename2 ...

The following options are supported:

- n number The first number lines of each input file will be copied to standard output. The number option-argument must be a positive decimal integer.

- number The number argument is a positive decimal integer with the same effect as the -n number option.

If no options are specified, head will act as if -n 10 had been specified.

The following operands are supported:

file A path name of an input file. If no file operands are specified, the standard input will be used.

See largefile(5) for the description of the behavior of head when encountering files greater than or equal to 2 Gbyte (231 bytes).

EXAMPLES EXAMPLE 1 Writing the first ten lines of all files

To write the first ten lines of all files (except those with a leading period) in the directory:

example% head *

See environ(5) for descriptions of the following environment variables that affect the execution of head: LC_CTYPE, LC_MESSAGES, and NLSPATH.

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

See attributes(5) for descriptions of the following attributes:
head(1)

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

**SEE ALSO**
cat(1), more(1), pg(1), tail(1), attributes(5), environ(5), largefile(5)
NAME

history, fc – process command history list

SYNOPSIS

/usr/bin/fc [first [last]]

/usr/bin/fc -l [-nr] [first [last]]

/usr/bin/fc -s [old = new] [first]

The fc utility lists or edits and reexecutes, commands previously entered to an interactive sh.

The command history list references commands by number. The first number in the list is selected arbitrarily. The relationship of a number to its command will not change except when the user logs in and no other process is accessing the list, at which time the system may reset the numbering to start the oldest retained command at another number (usually 1). When the number reaches the value in HISTSIZE or 128 (whichever is greater), the shell may wrap the numbers, starting the next command with a lower number (usually 1). However, despite this optional wrapping of numbers, fc will maintain the time-ordering sequence of the commands. For example, if four commands in sequence are given the numbers 32 766, 32 767, 1 (wrapped), and 2 as they are executed, command 32 767 is considered the command previous to 1, even though its number is higher.

When commands are edited (when the -l option is not specified), the resulting lines will be entered at the end of the history list and then reexecuted by sh. The fc command that caused the editing will not be entered into the history list. If the editor returns a non-zero exit status, this will suppress the entry into the history list and the command reexecution. Any command-line variable assignments or redirection operators used with fc will affect both the fc command itself as well as the command that results, for example:

```
fc -s -- -- 1 2>/dev/null
```

reinvokes the previous command, suppressing standard error for both fc and the previous command.

Display the history list; if n is given, display only the n most recent events.

-r Reverse the order of printout to be most recent first rather than oldest first.

-h Display the history list without leading numbers. This is used to produce files suitable for sourcing using the -h option to the csh built-in command, source(1).

History Substitution
History substitution allows you to use words from previous command lines in the command line you are typing. This simplifies spelling corrections and the repetition of complicated commands or arguments. Command lines are saved in the history list, the size of which is controlled by the `history` variable. The `history` shell variable may be set to the maximum number of command lines that will be saved in the history file; i.e.:

```bash
set history = 200
```

will allow the history list to keep track of the most recent 200 command lines. If not set, the C shell saves only the most recent command.

A history substitution begins with a `!` (although you can change this with the `histchars` variable) and may occur anywhere on the command line; history substitutions do not nest. The `!` can be escaped with \ to suppress its special meaning.

Input lines containing history substitutions are echoed on the terminal after being expanded, but before any other substitutions take place or the command gets executed.

**Event Designators:**

An event designator is a reference to a command line entry in the history list.

- `!` Start a history substitution, except when followed by a space character, tab, newline, = or (.
- `!!` Refer to the previous command. By itself, this substitution repeats the previous command.
- `!n` Refer to command line n.
- `!-n` Refer to the current command line minus n.
- `!str` Refer to the most recent command starting with str.
- `!?str?` Refer to the most recent command containing str.
- `!?str? additional` Refer to the most recent command containing str and append additional to that referenced command.
- `!{command} additional` Refer to the most recent command beginning with command and append additional to that referenced command.
- `^previous_word^replacement^` Repeat the previous command line replacing the string `previous_word` with the string `replacement`. This is equivalent to the history substitution.
To re-execute a specific previous command AND make such a substitution, say, re-executing command #6,

To re-execute a specific previous command, say, re-executing command #6,

Word Designators:
A ‘:’ (colon) separates the event specification from the word designator. It can be omitted if the word designator begins with a ^, $, *, – or %. If the word is to be selected from the previous command, the second ! character can be omitted from the event specification. For instance, !!!:1 and !!:1 both refer to the first word of the previous command, while !!!$ and !!$ both refer to the last word in the previous command. Word designators include:

# The entire command line typed so far.
0 The first input word (command).
n The n’th argument.
^ The first argument, that is, 1.
$ The last argument.
% The word matched by (the most recent) ?s search.
x−y A range of words; −y abbreviates 0−y.
* All the arguments, or a null value if there is just one word in the event.
x* Abbreviates x−$.
x− Like x* but omitting word $.

Modifiers:
After the optional word designator, you can add a sequence of one or more of the following modifiers, each preceded by a :.

h Remove a trailing pathname component, leaving the head.
x Remove a trailing suffix of the form ‘. xxx’, leaving the basename.
e Remove all but the suffix, leaving the extension.
\oldchars//replacements/ Substitute \replacements for \oldchars. \oldchars is a string that may contain embedded blank spaces, whereas \previous_word in the event designator

User Commands 529
Remove all leading pathname components, leaving the tail.

&
Repeat the previous substitution.

g
Apply the change to the first occurrence of a match in each word, by prefixing the above (for example, g&).

p
Print the new command but do not execute it.

q
Quote the substituted words, escaping further substitutions.

x
Like q, but break into words at each space character, tab or newline.

Unless preceded by a g, the modification is applied only to the first string that matches oldchars; an error results if no string matches.

The left-hand side of substitutions are not regular expressions, but character strings. Any character can be used as the delimiter in place of / . A backslash quotes the delimiter character. The character & , in the right hand side, is replaced by the text from the left-hand-side. The & can be quoted with a backslash. A null oldchars uses the previous string either from a oldchars or from a contextual scan string s from ! ? s . You can omit the rightmost delimiter if a newline immediately follows replacements; the rightmost ? in a context scan can similarly be omitted.

Without an event specification, a history reference refers either to the previous command, or to a previous history reference on the command line (if any).

Using fc, in the form of

fc -e [ old=new ] [ command ], the command is re-executed after the substitution old=new is performed. If there is not a command argument, the most recent command typed at this terminal is executed.

Using fc in the form of

fc [ -e ename ] [ -nlx ] [ first [ last ] ] a range of commands from first to last is selected from the last HISTSIZE commands that were typed at the terminal. The arguments first and last may be specified as a number or as a string. A string is used to locate the most recent command starting with the given string. A negative number is used as an offset to the current command number. If the -l flag is selected, the commands are listed on standard output. Otherwise, the editor program -e ename is invoked on a file containing these keyboard commands. If ename is not supplied, then the value of the
variable FCEDIT (default /bin/ed) is used as the editor. When editing is complete, the edited command(s) is executed. If last is not specified then it will be set to first. If first is not specified the default is the previous command for editing and -16 for listing. The flag -r reverses the order of the commands and the flag -n suppresses command numbers when listing. (See ksh(1) for more about command line editing.)

HISTFILE
If this variable is set when the shell is invoked, then the value is the pathname of the file that will be used to store the command history.

HISTSIZE
If this variable is set when the shell is invoked, then the number of previously entered commands that are accessible by this shell will be greater than or equal to this number. The default is 128.

Command Re-entry:
The text of the last HISTSIZE (default 128) commands entered from a terminal device is saved in a history file. The file $HOME/.sh_history is used if the HISTFILE variable is not set or if the file it names is not writable. A shell can access the commands of all interactive shells which use the same named HISTFILE. The special command fc is used to list or edit a portion of this file. The portion of the file to be edited or listed can be selected by number or by giving the first character or characters of the command. A single command or range of commands can be specified. If you do not specify an editor program as an argument to fc then the value of the variable FCEDIT is used. If FCEDIT is not defined then /bin/ed is used. The edited command(s) is printed and re-executed upon leaving the editor. The editor name – is used to skip the editing phase and to re-execute the command. In this case a substitution parameter of the form old=new can be used to modify the command before execution. For example, if r is aliased to `fc -e ` then typing `r bad=good c` will re-execute the most recent command which starts with the letter c, replacing the first occurrence of the string bad with the string good.

Using the fc built-in command within a compound command will cause the whole command to disappear from the history file.

OPTIONS
The following options are supported:

-e editor   Use the editor named by editor to edit the commands. The editor string is a utility name, subject to search via the PATH variable. The value in the FCEDIT variable is used as a default when -e is not specified. If FCEDIT is null or unset, ed will be used as the editor.

-l   (The letter ell.) List the commands rather than invoking an editor on them. The commands will be written in the sequence indicated by the first and last operands, as affected by -r, with each command preceded by the command number.

-n   Suppress command numbers when listing with -l.
-r  Reverse the order of the commands listed (with -l) or edited (with neither -l nor -s).

-s  Re-execute the command without invoking an editor.

**OPERANDS**

The following operands are supported:

- **first**
- **last**

Select the commands to list or edit. The number of previous commands that can be accessed is determined by the value of the **HISTSIZE** variable. The value of **first** or **last** or both will be one of the following:

- **[+]number**
  A positive number representing a command number; command numbers can be displayed with the -l option.

- **-number**
  A negative decimal number representing the command that was executed **number** of commands previously. For example, -1 is the immediately previous command.

- **string**
  A string indicating the most recently entered command that begins with that string. If the **old=new** operand is not also specified with -s, the string form of the **first** operand cannot contain an embedded equal sign.

When the synopsis form with -s is used:

- If **first** is omitted, the previous command will be used. For the synopsis forms without -s:
  - If **last** is omitted, **last** defaults to the previous command when -l is specified; otherwise, it defaults to **first**.
  - If **first** and **last** are both omitted, the previous 16 commands will be listed or the previous single command will be edited (based on the -l option).
  - If **first** and **last** are both present, all of the commands from **first** to **last** will be edited (without -l) or listed (with -l). Editing multiple commands will be accomplished by presenting to the editor all of the commands at one time, each command starting on a new line. If **first** represents a newer command than **last**, the commands will be listed or edited in reverse sequence, equivalent to using -r. For example, the following commands on the first line are equivalent to the corresponding commands on the second:

```
fc  -r 10 20  fc  30 40
fc  20 10  fc  -r 40 30
```
When a range of commands is used, it will not be an error to specify first or last values that are not in the history list; fc will substitute the value representing the oldest or newest command in the list, as appropriate. For example, if there are only ten commands in the history list, numbered 1 to 10:

fc -l
fc 1 99 will list and edit, respectively, all ten commands.

old=new Replace the first occurrence of string old in the commands to be reexecuted by the string new.

OUTPUT When the -l option is used to list commands, the format of each command in the list is as follows:

"%d\t%s\n", <line number>, <command>

If both the -l and -n options are specified, the format of each command is:

"\t%s\n", <command>

If the command consists of more than one line, the lines after the first are displayed as:

"\t%s\n", <continued-command>

EXAMPLES

EXAMPLE 1 Using history and fc

<table>
<thead>
<tr>
<th>csh</th>
<th>ksh</th>
</tr>
</thead>
<tbody>
<tr>
<td>% history</td>
<td>$ fc -l</td>
</tr>
<tr>
<td>1 cd /etc</td>
<td>1 cd /etc</td>
</tr>
<tr>
<td>2 vi passwd</td>
<td>2 vi passwd</td>
</tr>
<tr>
<td>3 date</td>
<td>3 date</td>
</tr>
<tr>
<td>4 cd</td>
<td>4 cd</td>
</tr>
<tr>
<td>5 du .</td>
<td>5 du</td>
</tr>
<tr>
<td>6 ls -t</td>
<td>6 ls -t</td>
</tr>
<tr>
<td>7 history</td>
<td>7 fc -l</td>
</tr>
<tr>
<td>% !d</td>
<td>$ fc -e - d</td>
</tr>
<tr>
<td>du .</td>
<td>du .</td>
</tr>
<tr>
<td>262 ./SCCS</td>
<td>262 ./SCCS</td>
</tr>
<tr>
<td>336 .</td>
<td>336 .</td>
</tr>
<tr>
<td>% !da</td>
<td>$ fc -e - da</td>
</tr>
<tr>
<td>%</td>
<td>$ alias !=&quot;/fc -e -&quot;</td>
</tr>
</tbody>
</table>
EXAMPLE 1 Using history and fc (Continued)

% !!

date
Thu Jul 21 17:29:56 PDT 1994

! alias fc -e '-'

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of fc: LC_CTYPE, LC_MESSAGES, and NLSPATH.

FCEDIT This variable, when expanded by the shell, determines the default value for the e editor option’s editor option-argument. If FCEDIT is null or unset, ed will be used as the editor.

HISTFILE Determine a pathname naming a command history file. If the HISTFILE variable is not set, the shell may attempt to access or create a file .sh_history in the user’s home directory. If the shell cannot obtain both read and write access to, or create, the history file, it will use an unspecified mechanism that allows the history to operate properly. (References to history “file” in this section are understood to mean this unspecified mechanism in such cases.) fc may choose to access this variable only when initializing the history file; this initialization will occur when fc or sh first attempt to retrieve entries from, or add entries to, the file, as the result of commands issued by the user, the file named by the ENV variable, or a system startup file such as /etc/profile. (The initialization process for the history file can be dependent on the system startup files, in that they may contain commands that will effectively preempt the user’s settings of HISTFILE and HISTSIZE. For example, function definition commands are recorded in the history file, unless the set -o nolog option is set. If the system administrator includes function definitions in some system startup file called before the ENV file, the history file will be initialized before the user gets a chance to influence its characteristics.) The variable HISTFILE is accessed initially when the shell is invoked. Any changes to HISTFILE will not take effect until another shell is invoked.

HISTSIZE Determine a decimal number representing the limit to the number of previous commands that are accessible. If this variable is unset, an unspecified default greater than or equal to 128 will be used. The variable HISTSIZE is accessed initially when the shell is invoked. Any changes to HISTSIZE will not take effect until another shell is invoked.

EXIT STATUS The following exit values are returned:

0 Successful completion of the listing.

>0 An error occurred.
Otherwise, the exit status will be that of the commands executed by fc.

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

**SEE ALSO**

csh(1), ed(1), ksh(1), set(1), set(1F), sh(1), source(1), attributes(5), environ(5)
hostid(1)

NAME  hostid – print the numeric identifier of the current host

SYNOPSIS  /usr/bin/hostid

DESCRIPTION  The hostid command prints the identifier of the current host in hexadecimal. This numeric value is likely to differ when hostid is run on a different machine.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO  sysinfo(2), gethostid(3C), attributes(5)
NAME
hostname – set or print name of current host system

SYNOPSIS
/usr/bin/hostname [name-of-host]

DESCRIPTION
The hostname command prints the name of the current host, as given before the login prompt. The super-user can set the hostname by giving an argument.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO
uname(1), attributes(5)
iconv(1)

NAME    iconv – code set conversion utility

SYNOPSIS    iconv -f fromcode -t tocode [file...]

DESCRIPTION    The iconv utility converts the characters or sequences of characters in file from one
code set to another and writes the results to standard output. If no conversion exists
for a particular character, it is converted to the underscore _ in the target code set.

The list of supported conversions and the locations of the associated conversion tables
are provided in the iconv(5) manual page.

OPTIONS    The following options are supported:
    -f fromcode    Identifies the input code set.
    -t tocode      Identifies the output code set.

OPERANDS    The following operands are supported:
    file          A path name of the input file to be translated. If file is omitted, the
                   standard input is used.

EXAMPLES    EXAMPLE 1 Converting and storing files

The following example converts the contents of file mail1 from code set 8859 to
646fr and stores the results in file mail.local:

    example% iconv -f 8859 -t 646fr mail1 > mail.local

ENVIRONMENT VARIABLES    See environ(5) for descriptions of the following environment variables that affect the
execution of iconv: LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS    The following exit values are returned:
    0        Successful completion.
    1        An error has occurred.

FILES    /usr/lib/iconv/*.so
         conversion modules
    /usr/lib/iconv/*.t
         conversion tables
    /usr/lib/iconv/iconv_data
         list of conversions supported by conversion tables
    /usr/lib/iconv/geniconvtbl/binarytables/*.bt
         conversion binary tables

ATTRIBUTES    See attributes(5) for descriptions of the following attributes:
The `iconv` utility can use conversion modules (`/usr/lib/iconv/*.so`), conversion tables (`/usr/lib/iconv/*.t`), or conversion binary tables (`/usr/lib/iconv/geniconvtbl/binarytables/*.bt`) to do the code set conversion. The `iconv` utility uses `iconv_open(3C)` to see if a particular code set conversion is available in the `iconv(3C)` function. `iconv_open(3C)` first tries to find out if there is a conversion binary table and then if there is a conversion module. If neither is available in your system, `iconv_open(3C)` will return a failure code. Then, finally, `iconv` will search for a conversion table.

Refer to the `/usr/share/man/man5/iconv_locale.5` manual page in the Asian localized releases for information on which code set conversions are supported. For example, the command

```
example% man -s 5 iconv_ja
```

would display the manual page describing the code set conversions that are supported for the Japanese locale.

Notice that the `iconv_locale.5` manual page may not exist in your system, depending on which locale you chose to install during the system installation.
The `indicator` function displays application specific alarms or the "working" indicator, or both, on the FMLI banner line. The argument `string` is a string to be displayed on the banner line, and should always be the last argument given. Note that `string` is not automatically cleared from the banner line.

- `bn` The `-b` option rings the terminal bell `n` times, where `n` is an integer from 1 to 10. The default value is 1. If the terminal has no bell, the screen is flashed instead, if possible.

- `-c column` The `-c` option defines the column of the banner line at which to start the indicator string. The argument `column` must be an integer from 0 to `DISPLAYW-1`. If the `-c` option is not used, `column` defaults to 0.

- `-l length` The `-l` option defines the maximum length of the string displayed. If `string` is longer than `length` characters, it will be truncated. The argument `length` must be an integer from 1 to `DISPLAYW`. If the `-l` option is not used, `length` defaults to `DISPLAYW`. Note that if `string` doesn't fit it will be truncated.

- `-o` The `-o` option causes `indicator` to duplicate its output to `stdout`.

- `-w` The `-w` option turns on the "working" indicator.

**EXAMPLE 1** A sample output of the `indicator` command.

When the value entered in a form field is invalid, the following use of `indicator` will ring the bell three times and display the word `WRONG` starting at column 1 of the banner line.

```bash
invalidmsg='indicator -b 3 -c 1 "WRONG"'
```

To clear the indicator after telling the user the entry is wrong:

```bash
invalidmsg='indicator -b 9 -c 1 "WRONG"; sleep 3;
indicator -c 1 " "
```

In this example the value of `invalidmsg` (in this case the default value Input is not valid), still appears on the FMLI message line.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

**SEE ALSO** attributes(5)
NAME
indxbib – create an inverted index to a bibliographic database

SYNOPSIS
indxbib database-file...

DESCRIPTION
indxbib makes an inverted index to the named database-file (which must reside
within the current directory), typically for use by lookbib(1) and refer(1). A
database contains bibliographic references (or other kinds of information) separated by
blank lines.

A bibliographic reference is a set of lines, constituting fields of bibliographic
information. Each field starts on a line beginning with a ‘%’, followed by a key-letter,
then a blank, and finally the contents of the field, which may continue until the next
line starting with ‘%’.

indxbib is a shell script that calls two programs: /usr/lib/refer/mkey and
/usr/lib/refer/inv. mkey truncates words to 6 characters, and maps upper case
to lower case. It also discards words shorter than 3 characters, words among the 100
most common English words, and numbers (dates) < 1000 or > 2099. These parameters
can be changed.

indxbib creates an entry file (with a .ia suffix), a posting file (.ib), and a tag file
(.ic), in the working directory.

FILES
/usr/lib/refer/mkey
/usr/lib/refer/inv
x.ia entry file
x.ib posting file
x.ic tag file
x.ig reference file

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWdoc</td>
</tr>
</tbody>
</table>

SEE ALSO
addbib(1), lookbib(1), refer(1), roffbib(1), sortbib(1), attributes(5)

BUGS
All dates should probably be indexed, since many disciplines refer to literature written
in the 1800s or earlier.

indxbib does not recognize pathnames.
install(1B)

NAME
install – install files

SYNOPSIS
/usr/ucb/install [-cs] [-g group] [-m mode] [-o owner] filename1

/usr/ucb/install [-cs] [-g group] [-m mode] [-o owner] filename2
directory

/usr/ucb/install [-d] [-g group] [-m mode] [-o owner] directory

DESCRIPTION
install is used within makefiles to copy new versions of files into a destination
directory and to create the destination directory itself.

The first two forms are similar to the cp(1) command with the addition that executable
files can be stripped during the copy and the owner, group, and mode of the installed
file(s) can be given.

The third form can be used to create a destination directory with the required owner,
group and permissions.

Note: install uses no special privileges to copy files from one place to another. The
implications of this are:

■ You must have permission to read the files to be installed.
■ You must have permission to copy into the destination file or directory.
■ You must have permission to change the modes on the final copy of the file if you
  want to use the -m option to change modes.
■ You must be superuser if you want to specify the ownership of the installed file
  with -o. If you are not the super-user, or if -o is not in effect, the installed file
  will be owned by you, regardless of who owns the original.

OPTIONS
  -c Copy files. In fact install always copies files, but the -c option is
  retained for backwards compatibility with old shell scripts that
  might otherwise break.

  -d Create a directory. Missing parent directories are created as
  required as in mkdir -p. If the directory already exists, the owner,
group and mode will be set to the values given on the command
  line.

  -s Strip executable files as they are copied.

  -g group Set the group ownership of the installed file or directory. (staff by
default.)

  -m mode Set the mode for the installed file or directory. (0755 by default.)

  -o owner If run as root, set the ownership of the installed file to the user-ID
  of owner.

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

chgrp(1), chmod(1), chown(1), cp(1), mkdir(1), strip(1), install(1M), attributes(5)
NAME
ipcrm – remove a message queue, semaphore set, or shared memory ID

SYNOPSIS
ipcrm [-m shmid] [-q msqid] [-s semid] [-M shmkey] [-Q msgkey]
       [-S semkey]

DESCRIPTION
ipcrm removes one or more messages, semaphores, or shared memory identifiers.

OPTIONS
The identifiers are specified by the following options:

- `-m shmid`
  Remove the shared memory identifier `shmid` from the system. The shared memory
  segment and data structure associated with it are destroyed after the last detach.

- `-q msqid`
  Remove the message queue identifier `msqid` from the system and destroy the
  message queue and data structure associated with it.

- `-s semid`
  Remove the semaphore identifier `semid` from the system and destroy the set of
  semaphores and data structure associated with it.

- `-M shmkey`
  Remove the shared memory identifier, created with key `shmkey`, from the system.
  The shared memory segment and data structure associated with it are destroyed after
  the last detach.

- `-Q msgkey`
  Remove the message queue identifier, created with key `msgkey`, from the system and
  destroy the message queue and data structure associated with it.

- `-S semkey`
  Remove the semaphore identifier, created with key `semkey`, from the system and
  destroy the set of semaphores and data structure associated with it.

The details of the removes are described in msgctl(2), shmctl(2), and semctl(2).
Use the `ipcs` command to find the identifiers and keys.

ENVIRONMENT VARIABLES
See environ(5) for descriptions of the following environment variables that affect the
execution of ipcrm: LANG, LC_ALL, LC_CTYPE, LC_MESSAGES, and NLSPATH.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWipc</td>
</tr>
</tbody>
</table>

SEE ALSO
ipcs(1), msgctl(2), msgget(2), msgsnd(2), msgctlv(2), semget(2), semctl(2),
semop(2), shmctl(2), shmat(2), shmget(2), shmop(2), attributes(5), environ(5)
NAME
	ipcs – report inter-process communication facilities status
SYNOPSIS
	ipcs [-AAbcimopqst] [-D mtype]
DESCRIPTION
The ipcs utility prints information about active inter-process communication facilities. The information that is displayed is controlled by the options supplied. Without options, information is printed in short format for message queues, shared memory, and semaphores that are currently active in the system.

OPTIONS
The following options are supported:

-m
Prints information about active shared memory segments.

-q
Prints information about active message queues.

-s
Prints information about active semaphores.

If -m, -q, or -s are specified, information about only those indicated is printed. If none of these three is specified, information about all three is printed subject to these options:

-a
Uses all XCU5 print options. (This is a shorthand notation for -b, -c, -o, -p, and -t.)

-A
Uses all print options. (This is a shorthand notation for -b, -c, -i, -o, -p, and -t.)

-b
Prints information on biggest allowable size: maximum number of bytes in messages on queue for message queues, size of segments for shared memory, and number of semaphores in each set for semaphores. See below for meaning of columns in a listing.

-c
Prints creator’s login name and group name. See below.

-D mtype
Displays, in hexadecimal and ASCII, the contents of all messages of type mtype found on any message queue that the user invoking ipcs has permission to read. If mtype is 0, all messages are displayed. If mtype is negative, all messages with type less than or equal to the absolute value of mtype are displayed. (See msgrcv(2) and msgsnd(2)).

-i
Prints number of ISM attaches to shared memory segments.

-o
Prints information on outstanding usage: number of messages on queue and total number of bytes in messages on queue for message queues and number of processes attached to shared memory segments.

-p
Prints process number information: process ID of last process to send a message, process ID of last process to receive a message on message queues, process ID of creating process, and process ID of last process to attach or detach on shared memory segments. See below.
Prints time information: time of the last control operation that changed the access permissions for all facilities, time of last `msgsnd(2)` and last `msgrcv(2)` on message queues, time of last `shmat(2)` and last `shmdt(2)` on shared memory (see `shmop(2)`), time of last `semop(2)` on semaphores. See below.

The column headings and the meaning of the columns in an `ipcs` listing are given below. The letters in parentheses indicate the options that cause the corresponding heading to appear. “all” means that the heading always appears. **Note:** These options only determine what information is provided for each facility; they do not determine which facilities are listed.

<table>
<thead>
<tr>
<th>T (all)</th>
<th>Type of the facility:</th>
</tr>
</thead>
<tbody>
<tr>
<td>q</td>
<td>message queue</td>
</tr>
<tr>
<td>m</td>
<td>shared memory segment</td>
</tr>
<tr>
<td>s</td>
<td>semaphore</td>
</tr>
</tbody>
</table>

| ID (all) | The identifier for the facility entry. |
| KEY (all) | The key used as an argument to `msgget(2)`, `semget(2)`, or `shmget(2)` to create the facility entry. **(Note:** The key of a shared memory segment is changed to `IPC_PRIVATE` when the segment has been removed until all processes attached to the segment detach it.) |

<table>
<thead>
<tr>
<th>MODE (all)</th>
<th>The facility access modes and flags: The mode consists of 11 characters that are interpreted as follows. The first two characters are:</th>
</tr>
</thead>
</table>
| R           | A process is waiting on a `msgrcv(2)`.
| S           | A process is waiting on a `msgsnd(2)`.
| D           | The associated shared memory segment has been removed. It will disappear when the last process attached to the segment detaches it. **(Note:** If the shared memory segment identifier is removed via an `IPC_RMID` call to `shmct1(2)` before the process has detached from the segment with `shmdt(2)`, the segment is no longer visible to `ipcs` and it will not appear in the `ipcs` output.) |
| C           | The associated shared memory segment is to be cleared when the first attach is executed. |
| -           | The corresponding special flag is not set. |
The next nine characters are interpreted as three sets of three bits each. The first set refers to the owner’s permissions; the next to permissions of others in the user-group of the facility entry; and the last to all others. Within each set, the first character indicates permission to read, the second character indicates permission to write or alter the facility entry, and the last character is currently unused.

The permissions are indicated as follows:

- **r**  Read permission is granted.
- **w**  Write permission is granted.
- **a**  Alter permission is granted.
- **−**  The indicated permission is not granted.

**OWNER** (all)  The login name of the owner of the facility entry.

**GROUP** (all)  The group name of the group of the owner of the facility entry.

**CREATOR** (a,A,c)  The login name of the creator of the facility entry.

**CGROUP** (a,A,c)  The group name of the group of the creator of the facility entry.

**CBYTES** (a,A,o)  The number of bytes in messages currently outstanding on the associated message queue.

**QNUM** (a,A,o)  The number of messages currently outstanding on the associated message queue.

**QBYTES** (a,A,b)  The maximum number of bytes allowed in messages outstanding on the associated message queue.

**LSPID** (a,A,p)  The process ID of the last process to send a message to the associated queue.

**LRPID** (a,A,p)  The process ID of the last process to receive a message from the associated queue.

**STIME** (a,A,t)  The time the last message was sent to the associated queue.

**RTIME** (a,A,t)  The time the last message was received from the associated queue.

**CTIME** (a,A,t)  The time when the associated entry was created or changed.

**ISMATTCH** (a,i)  The number of ISM attaches to the associated shared memory segments.
NATTC (a,A,o) The number of processes attached to the associated shared memory segment.

SEGSZ (a,A,b) The size of the associated shared memory segment.

CPID (a,A,p) The process ID of the creator of the shared memory entry.

LPID (a,A,p) The process ID of the last process to attach or detach the shared memory segment.

ATIME (a,A,t) The time the last attach was completed to the associated shared memory segment.

DTIME (a,A,t) The time the last detach was completed on the associated shared memory segment.

NSEMS (a,A,b) The number of semaphores in the set associated with the semaphore entry.

OTIME (a,A,t) The time the last semaphore operation was completed on the set associated with the semaphore entry.

See environ(5) for descriptions of the following environment variables that affect the execution of ipcs: LANG, LC_ALL, LC_CTYPE, LC_MESSAGES, and NLSPATH.

TZ Determine the timezone for the time strings written by ipcs.

FILES
/etc/group group names
/etc/passwd user names

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWipc</td>
</tr>
</tbody>
</table>

SEE ALSO
ipcrm(1), msgget(2), msgsids(2), msgrcv(2), msgsnd(2), semget(2), semids(2), semop(2), shmtl(2), shmget(2), shmids(2), shmop(2), attributes(5), environ(5)

NOTES
Things can change while ipcs is running. The information it gives is guaranteed to be accurate only when it was retrieved.
**NAME**
isainfo – describe instruction set architectures

**SYNOPSIS**
isainfo [-v] [-b | -n | -k]

**DESCRIPTION**
The *isainfo* utility is used to identify various attributes of the instruction set architectures supported on the currently running system. Among the questions it can answer are whether 64-bit applications are supported, or whether the running kernel uses 32-bit or 64-bit device drivers.

When invoked with no flags, *isainfo* prints the name(s) of the native instruction sets for applications supported by the current version of the operating system. These will be a subset of the list returned by *isalist*(1). The subset corresponds to the basic applications environments supported by the currently running system.

**OPTIONS**
The following options are supported:

- `-b` Prints the number of bits in the address space of the native instruction set.
- `-k` Prints the name of the instruction set(s) used by the operating system kernel components such as device drivers and STREAMS modules.
- `-n` Prints the name of the native instruction set used by portable applications supported by the current version of the operating system.
- `-v` Prints more detailed information about the other options.

**EXAMPLES**

**EXAMPLE 1** Invoking isainfo on a 32-bit IA platform

```bash
example$ isainfo -v
32-bit i386 applications
example$ isainfo -k
i386
```

**EXAMPLE 2** Invoking isainfo on a system running the 32-bit operating system on a 64-bit SPARC processor

```bash
example$ isainfo -n sparc
exmaple$ isainfo -v
32-bit sparc applications
example$ isainfo -k
32-bit sparc kernel modules
```

**EXAMPLE 3** Invoking isainfo on the same hardware platform (that is, a 64-bit SPARC processor) running the 64-bit operating system

```bash
example$ isainfo
sparcv9 sparc
example$ isainfo -n
sparcv9
example$ isainfo -v
64-bit sparcv9 applications 32-bit sparc applications
example$ isainfo -vk
64-bit sparcv9 kernel modules
```
EXAMPLE 3 Invoking isainfo on the same hardware platform (that is, a 64-bit SPARC processor) running the 64-bit operating system  (Continued)

EXIT STATUS
Non-zero Flags are not specified correctly, or the command is unable to recognize attributes of the system on which it is running. An error message is printed to stderr.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO
isalist(1), uname(1), psrinfo(1M), sysinfo(2), attributes(5), isalist(5)
isalist – display the native instruction sets executable on this platform

isalist

isalist prints the names of the native instruction sets executable on this platform on the standard output, as returned by the SI_ISALIST command of sysinfo(2).

The names are space-separated and are ordered in the sense of best performance. That is, earlier-named instruction sets may contain more instructions than later-named instruction sets; a program that is compiled for an earlier-named instruction sets will most likely run faster on this machine than the same program compiled for a later-named instruction set.

Programs compiled for instruction sets that do not appear in the list will most likely experience performance degradation or not run at all on this machine.

The instruction set names known to the system are listed in isalist(5). These names may or may not match predefined names or compiler options in the C language compilation system.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO

optisa(1), uname(1), sysinfo(2), attributes(5), isalist(5)
When Job Control is enabled, the Bourne shell built-in `jobs` reports all jobs that are stopped or executing in the background. If `job_id` is omitted, all jobs that are stopped or running in the background will be reported. The following options will modify/enhance the output of `jobs`:

- `-l` Reports the process group ID and working directory of the jobs.
- `-p` Reports only the process group ID of the jobs.
- `-x` Replaces any `job_id` found in `command` or `arguments` with the corresponding process group ID, and then executes `command` passing it `arguments`.

When the shell is invoked as `jsh`, Job Control is enabled in addition to all of the functionality described previously for `sh`. Typically Job Control is enabled for the interactive shell only. Non-interactive shells typically do not benefit from the added functionality of Job Control.

With Job Control enabled every command or pipeline the user enters at the terminal is called a `job_id`. All jobs exist in one of the following states: foreground, background or stopped. These terms are defined as follows:

1. A job in the **foreground** has read and write access to the controlling terminal.
2. A job in the background is denied read access and has conditional write access to the controlling terminal (see `attty(1)`)

3. A stopped job is a job that has been placed in a suspended state, usually as a result of a SIGTSTP signal (see `signal(3HEAD)`).

Every job that the shell starts is assigned a positive integer, called a job_id number which is tracked by the shell and will be used as an identifier to indicate a specific job. Additionally, the shell keeps track of the current and previous jobs. The current job is the most recent job to be started or restarted. The previous job is the first non-current job.

The acceptable syntax for a Job Identifier is of the form:

```
%job_id
```

where job_id may be specified in any of the following formats:

- `%` or `+` for the current job
- `-` for the previous job
- `?<string>` to specify the job for which the command line uniquely contains `string`.
- `n` for job number `n`, where `n` is a job number
- `pref` where `pref` is a unique prefix of the command name (for example, if the command `ls -l name` were running in the background, it could be referred to as `%ls`; `pref` cannot contain blanks unless it is quoted.

When Job Control is enabled, `fg` resumes the execution of a stopped job in the foreground, also moves an executing background job into the foreground. If `%job_id` is omitted the current job is assumed.

When Job Control is enabled, `bg` resumes the execution of a stopped job in the background. If `%job_id` is omitted the current job is assumed.

`stop` stops the execution of a background job(s) by using its `job_id`, or of any process by using its `pid`; see `ps(1)`.

The C shell built-in, `jobs`, without an argument, lists the active jobs under job control.

```
csh -l
```

List process IDs, in addition to the normal information.

The shell associates a numbered `job_id` with each command sequence to keep track of those commands that are running in the background or have been stopped with `TSTP` signals (typically Control-Z). When a command or command sequence (semicolon-separated list) is started in the background using the `&` metacharacter, the shell displays a line with the job number in brackets and a list of associated process numbers:
To see the current list of jobs, use the `jobs` built-in command. The job most recently stopped (or put into the background if none are stopped) is referred to as the **current job** and is indicated with a ‘+’. The previous job is indicated with a ‘−’; when the current job is terminated or moved to the foreground, this job takes its place (becomes the new current job).

To manipulate jobs, refer to the `bg`, `fg`, `kill`, `stop`, and `%` built-in commands.

A reference to a job begins with a ‘%’. By itself, the percent sign refers to the current job. 

- `% %+ %` The current job.
- `%−` The previous job.
- `%j` Refer to job j as in: `kill -9 %j`. j can be a job number, or a string that uniquely specifies the command line by which it was started; `fg %vi` might bring a stopped vi job to the foreground, for instance.
- `%?string` Specify the job for which the command line uniquely contains `string`.

A job running in the background stops when it attempts to read from the terminal. Background jobs can normally produce output, but this can be suppressed using the 'stty tostop' command.

`fg` brings the current or specified `job_id` into the foreground.

`bg` runs the current or specified jobs in the background.

`stop` stops the execution of a background `job(s)` by using its `job_id`, or of any process by using its `pid`; see `ps(1)`.

`notify` will notify the user asynchronously when the status of the current job or specified jobs changes.

```
jobs(1)
```

The following options will modify/enhance the output of `jobs`:

- `-l` (The letter ell.) Provides more information about each job listed. This information includes the job number, current job, process group ID, state and the command that formed the job.
Displays only jobs that have stopped or exited since last notified.

Displays only the process IDs for the process group leaders of the selected jobs.

By default, `jobs` displays the status of all the stopped jobs, running background jobs, and all jobs whose status has changed and have not been reported by the shell.

If the monitor option of the `set` command is turned on, an interactive shell associates a job with each pipeline. It keeps a table of current jobs, printed by the `jobs` command, and assigns them small integer numbers. When a job is started asynchronously with &`, the shell prints a line which looks like:

```
[1] 1234
```

indicating that the job, which was started asynchronously, was job number 1 and had one (top-level) process, whose process id was 1234.

If you are running a job and wish to do something else you may hit the key `^Z` (Control-Z) which sends a `STOP` signal to the current job. The shell will then normally indicate that the job has been "Stopped" (see OUTPUT below), and print another prompt. You can then manipulate the state of this job, putting it in the background with the `bg` command, or run some other commands and then eventually bring the job back into the foreground with the foreground command `fg`. A `^Z` takes effect immediately and is like an interrupt, in that pending output and unread input are discarded when it is typed.

There are several ways to refer to jobs in the shell. A job can be referred to by the process id of any process of the job or by one of the following:

- `%number` The job with the given number.
- `%string` Any job whose command line begins with `string`; works only in the interactive mode when the history file is active.
- `%?string` Any job whose command line contains `string`; works only in the interactive mode when the history file is active.
- `%%` Current job.
- `%+` Equivalent to `%%`.
- `%-` Previous job.

The shell learns immediately whenever a process changes state. It normally informs you whenever a job becomes blocked so that no further progress is possible, but only just before it prints a prompt. This is done so that it does not otherwise disturb your work. When the monitor mode is on, each background job that completes triggers any trap set for `CHLD`. When you try to leave the shell while jobs are running or stopped,
you will be warned that 'You have stopped (running) jobs.' You may use the `jobs` command to see what they are. If you do this or immediately try to exit again, the shell will not warn you a second time, and the stopped jobs will be terminated.

`fg` will move a background job from the current environment into the foreground. Using `fg` to place a job in the foreground will remove its process ID from the list of those "known in the current shell execution environment." The `fg` command is available only on systems that support job control. If `job_id` is not specified, the current job is brought into the foreground.

`bg` resumes suspended jobs from the current environment by running them as background jobs. If the job specified by `job_id` is already a running background job, `bg` has no effect and will exit successfully. Using `bg` to place a job into the background causes its process ID to become "known in the current shell execution environment'", as if it had been started as an asynchronous list. The `bg` command is available only on systems that support job control. If `job_id` is not specified, the current job is placed in the background.

`stop` stops the execution of a background job(s) by using its `job_id`, or of any process by using its `pid`. See `ps(1)`.

**OUTPUT**

If the `-p` option is specified, the output consists of one line for each process ID:

```
"%d\n", "process ID"
```

Otherwise, if the `-l` option is not specified, the output is a series of lines of the form:

```
" [%d] %c %s %s\n", job-number, current, state, command
```

where the fields are as follows:

- **current**: The character + identifies the job that would be used as a default for the `fg` or `bg` commands. This job can also be specified using the `job_id` %+ or %+. The character - identifies the job that would become the default if the current default job were to exit; this job can also be specified using the `job_id` %-. For other jobs, this field is a space character. At most, one job can be identified with + and at most one job can be identified with -. If there is any suspended job, then the current job will be a suspended job. If there are at least two suspended jobs, then the previous job will also be a suspended job.

- **job-number**: A number that can be used to identify the process group to the `wait`, `fg`, `bg`, and `kill` utilities. Using these utilities, the job can be identified by prefixing the job number with %.

- **state**: One of the following strings (in the POSIX Locale):

```
"[RUN]", "SUSP"...
```
Running
Indicates that the job has not been suspended by a signal and has not exited.

Done
Indicates that the job completed and returned exit status zero.

Done(code)
Indicates that the job completed normally and that it exited with the specified non-zero exit status, code, expressed as a decimal number.

Stopped
Stopped(SIGTSTP)
Indicates that the job was suspended by the SIGTSTP signal.

Stopped(SIGSTOP)
Indicates that the job was suspended by the SIGSTOP signal.

Stopped(SIGTTIN)
Indicates that the job was suspended by the SIGTTIN signal.

Stopped(SIGTTOU)
Indicates that the job was suspended by the SIGTTOU signal.

The implementation may substitute the string Suspended in place of Stopped. If the job was terminated by a signal, the format of state is unspecified, but it will be visibly distinct from all of the other state formats shown here and will indicate the name or description of the signal causing the termination.

command
The associated command that was given to the shell.

If the -l option is specified, a field containing the process group ID is inserted before the state field. Also, more processes in a process group may be output on separate lines, using only the process ID and command fields.

ENVIRONMENT VARIABLES
See environ(5) for descriptions of the following environment variables that affect the execution of jobs, fg, and bg: LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS
The following exit values are returned for jobs, fg, and bg:

- 0  Successful completion.
- >0  An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>
SEE ALSO csh(1), kill(1), ksh(1), ps(1), sh(1), stop(1), shell_builtins(1), stty(1), wait(1), signal(3HEAD), attributes(5), environ(5)
### NAME
join – relational database operator

### SYNOPSIS

```bash
join [-a filenumber | -v filenumber] [-1 fieldnumber] [-2 fieldnumber]
       [-o list] [-e string] [-t char] file1 file2

join [-a filenumber] [-j fieldnumber] [-j1 fieldnumber] [-j2 fieldnumber]
       [-o list] [-e string] [-t char] file1 file2
```

### DESCRIPTION
The `join` command forms, on the standard output, a join of the two relations specified by the lines of `file1` and `file2`.

There is one line in the output for each pair of lines in `file1` and `file2` that have identical join fields. The output line normally consists of the common field, then the rest of the line from `file1`, then the rest of the line from `file2`. This format can be changed by using the `-o` option (see below). The `-a` option can be used to add unmatched lines to the output. The `-v` option can be used to output only unmatched lines.

The default input field separators are blank, tab, or new-line. In this case, multiple separators count as one field separator, and leading separators are ignored. The default output field separator is a blank.

If the input files are not in the appropriate collating sequence, the results are unspecified.

### OPTIONS
Some of the options below use the argument `filenumber`. This argument should be a 1 or a 2 referring to either `file1` or `file2`, respectively.

- `-a filenumber` In addition to the normal output, produce a line for each unpairable line in `file filenumber`, where `filenumber` is 1 or 2. If both `-a 1` and `-a 2` are specified, all unpairable lines will be output.

- `-e string` Replace empty output fields in the list selected by option `-o` with the string `string`.

- `-j fieldnumber` Equivalent to `-1fieldnumber -2fieldnumber`.

- `-j1 fieldnumber` Equivalent to `-1fieldnumber`.

- `-j2 fieldnumber` Equivalent to `-2fieldnumber`. Fields are numbered starting with 1.

- `-o list` Each output line includes the fields specified in `list`. Fields selected by `list` that do not appear in the input will be treated as empty output fields. (See the `-e` option.) Each element of which has the either the form `filenumber.fieldnumber`, or 0, which represents the `join` field. The common field is not printed unless specifically requested.

- `-t char` Use character `char` as a separator. Every appearance of `char` in a line is significant. The character `char` is used as the field separator for both input and output. With this

User Commands  559
option specified, the collating term should be the same as `sort` without the `-b` option.

-`v` `filenumber`

Instead of the default output, produce a line only for each unpairable line in `filenumber`, where `filenumber` is 1 or 2. If both `-v 1` and `-v 2` are specified, all unpairable lines will be output.

-`1` `fieldnumber`

Join on the `fieldnumber`th field of file 1. Fields are decimal integers starting with 1.

-`2` `fieldnumber`

Join on the `fieldnumber`th field of file 2. Fields are decimal integers starting with 1.

**OPERANDS**
The following operands are supported:

- `file1`
- `file2`

`file1` and `file2` must be sorted in increasing collating sequence as determined by `LC_COLLATE` on the fields on which they are to be joined, normally the first in each line (see `sort(1)`).

**USAGE**
See `largefile(5)` for the description of the behavior of `join` when encountering files greater than or equal to 2 Gbyte (2\(^31\) bytes).

**EXAMPLES**

**EXAMPLE 1** Joining the password file and group file

The following command line will join the password file and the group file, matching on the numeric group ID, and outputting the login name, the group name and the login directory. It is assumed that the files have been sorted in ASCII collating sequence on the group ID fields.

```
example% join -j1 4-j2 3 -o 1.1 2.1 1.6 -t:/etc/passwd /etc/group
```

**EXAMPLE 2** Using the `-o` option

The `-o` field essentially selects the union of the join fields. For example, given file `phone`:

<table>
<thead>
<tr>
<th>Name</th>
<th>Phone Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Don</td>
<td>+1 123-456-7890</td>
</tr>
<tr>
<td>Hal</td>
<td>+1 234-567-8901</td>
</tr>
<tr>
<td>Yasushi</td>
<td>+2 345-678-9012</td>
</tr>
</tbody>
</table>

and file `fax`:

<table>
<thead>
<tr>
<th>Name</th>
<th>Fax Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Don</td>
<td>+1 123-456-7899</td>
</tr>
</tbody>
</table>

560   man pages section 1: User Commands • Last Revised 8 Feb 2000
where the large expanses of white space are meant to each represent a single tab character), the command:

```bash
eexample join -t"tab" -a 1 -a 2 -e '(unknown)' -o 0,1.2,2.2 phone fax
```

would produce

<table>
<thead>
<tr>
<th>Name</th>
<th>Phone Number</th>
<th>Fax Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Don</td>
<td>+1 123-456-7890</td>
<td>+1 123-456-7899</td>
</tr>
<tr>
<td>Hal</td>
<td>+1 234-567-8901</td>
<td>(unknown)</td>
</tr>
<tr>
<td>Keith</td>
<td>(unknown)</td>
<td>+1 456-789-012</td>
</tr>
<tr>
<td>Yasushi</td>
<td>+2 345-678-9012</td>
<td>+2 345-678-9011</td>
</tr>
</tbody>
</table>

**ENVIRONMENT VARIABLES**

See `environ(5)` for descriptions of the following environment variables that affect the execution of `join`: `LC_CTYPE`, `LC_MESSAGES`, `LC_COLLATE`, and `NLSPATH`.

**EXIT STATUS**

The following exit values are returned:

- **0** All input files were output successfully.
- **>0** An error occurred.

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

**SEE ALSO**

`awk(1)`, `comm(1)`, `sort(1)`, `uniq(1)`, `attributes(5)`, `environ(5)`, `largefile(5)`

**NOTES**

With default field separation, the collating sequence is that of `sort -b`; with `-t`, the sequence is that of a plain sort.

The conventions of the `join`, `sort`, `comm`, `uniq`, and `awk` commands are wildly incongruous.
### NAME
kbd – manipulate the state of keyboard, or display the type of keyboard, or change the default keyboard abort sequence effect.

### SYNOPSIS
```
kbd [-r] [-t] [-a enable | disable | alternate] [-c on | off]
    [-d keyboard device]
```
```
kbd [-i] [-d keyboard device]
```

### DESCRIPTION
The kbd utility manipulates the state of the keyboard, or displays the keyboard type, or allows the default keyboard abort sequence effect to be changed. The abort sequence also applies to serial console devices. The kbd utility sets the `/dev/kbd` default keyboard device.

The `-i` option reads and processes default values for the keyclick and keyboard abort settings from the `/etc/default/kbd` keyboard default file. Only keyboards that support a clicker respond to the `-c` option. To turn clicking on by default, add or change the value of the `KEYCLICK` variable in the `/etc/default/kbd` file to:

```
KEYCLICK=on
```

Next, run the command `kbd -i` to change the setting. Valid settings for the `KEYCLICK` variable are `on` and `off`; all other values are ignored. If the `KEYCLICK` variable is not specified in the default file, the setting is unchanged.

The keyboard abort sequence effect (L1-A or STOP-A on the keyboard, and BREAK on the serial console input device on most systems) can only be changed by a superuser using the `-a` option. The system can be configured to ignore the keyboard abort sequence or trigger on the standard or alternate sequence.

A BREAK condition that originates from an erroneous electrical signal cannot be distinguished from one deliberately sent by remote DCE. As a remedy, use the `-a` option with Alternate Break to switch break interpretation. Due to the risk of incorrect sequence interpretation, binary protocols such as PPP, SLIP, and others should not be run over the serial console port when Alternate Break sequence is in effect. The Alternate Break sequence has no effect on the keyboard abort. For more information on the Alternate Break sequence, see `zap(7D), se(7D), and asy(7D)`.

On many systems, the default effect of the keyboard abort sequence is to suspend the operating system and enter the debugger or the monitor. Some systems feature key switches with a secure position. On these systems, setting the key switch to the `secure` position overrides any software default set with this command.

To permanently change the software default effect of the keyboard abort sequence, first add or change the value of the `KEYBOARD_ABORT` variable in the `/etc/default/kbd` file to:

```
KEYBOARD_ABORT=disable
```

Next, run the command `kbd -i` to change the setting. Valid settings are `enable`, `disable`, and `alternate`; all other values are ignored. If the variable is not specified in the default file, the setting is unchanged.
To set the abort sequence to the hardware BREAK, set the value of the KEYBOARD_ABORT variable in the /etc/default/kbd file to:

```
KEYBOARD_ABORT=enable
```

To change the current setting, run the command `kbd -i`. To set the abort sequence to the Alternate Break character sequence, first set the current value of the KEYBOARD_ABORT variable in the /etc/default/kbd file to:

```
KEYBOARD_ABORT=alternate
```

Next, run the command `kbd -i` to change the setting. When the Alternate Break sequence is in effect, only serial console devices are affected.

**OPTIONS**

The `kbd` utility supports the following options:

- `-i`
  Set keyboard defaults from the keyboard default file. With the exception of `-d keyboard device`, this option cannot be used with any other option. The `-i` option instructs the keyboard command to read and process keyclick and keyboard abort default values from the /etc/default/kbd file. The `-i` option can only be used by a superuser.

- `-r`
  Reset the keyboard as if power-up

- `-t`
  Return the type of the keyboard being used

- `-c on/off state`
  Turn the clicking of the keyboard on or off.
  - `on` Enable clicking
  - `off` Disable clicking

- `-a enable/disable/alternate state`
  Enable, disable, or alternate the keyboard abort sequence effect. By default, a keyboard abort sequence (typically Stop-A or L1-A on the keyboard and BREAK on the serial console device) suspends the operating system on most systems. The default keyboard behavior can be changed using this option. The `-a` option can only be used by a superuser.
  - `enable` Enable the default effect of the keyboard abort sequence (suspend the operating system and enter the debugger or the monitor)
  - `disable` Disable the default/alternate effect and ignore keyboard abort sequences
  - `alternate` Enable the alternate effect of the keyboard abort sequences (suspend the operating system and enter the debugger or the monitor) upon receiving the Alternate Break character sequence on the console. The Alternate Break sequence is defined by the drivers `zs(7D), se(7D), asy(7D)`. Due to a risk of incorrect sequence interpretation, binary protocols cannot be run over the serial console port when this value is used.
Specify the keyboard device being set. The default setting is "/dev/kbd".

**EXAMPLE 1** Displaying the keyboard type

To display the keyboard type:
```
example% kbd -t
```

Type 4 Sun keyboard
```
example% kbd -t
```

**EXAMPLE 2** Setting keyboard defaults

To set the keyboard defaults as specified in the keyboard default file:
```
example# kbd -i
```

FILES

/etc/rcS  shell script containing commands necessary to get the system to single-user mode
/dev/kbd  keyboard device file
/etc/default/kbd  keyboard default file containing software defaults for keyboard configurations.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>SPARC</td>
</tr>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO

loadkeys(1), kadb(1M), keytables(4), attributes(5), kb(7M), zs(7D), se(7D), asy(7D)

NOTES

Some server systems have key switches with a secure key position that can be read by system software. This key position overrides the normal default of the keyboard abort sequence effect and changes the default so the effect is disabled. When the key switch is in the secure position on these systems, the keyboard abort sequence effect cannot be overridden by the software default, which is settable with the kbd utility.

Currently, there is no way to determine the state of the keyboard click setting.
kdestroy – destroy Kerberos tickets

SYNOPSIS

```
/usr/bin/kdestroy [-q] [-c cache_name]
```

DESCRIPTION

The kdestroy utility destroys the user’s active Kerberos authorization tickets by writing zeros to the specified credentials cache that contains them. If the credentials cache is not specified, the default credentials cache is destroyed. If the credentials cache does not exist, kdestroy displays a message to that effect.

After overwriting the cache, kdestroy removes the cache from the system. The utility displays a message indicating the success or failure of the operation. If kdestroy is unable to destroy the cache, it will warn you by making your terminal beep.

If desired, you can place the kdestroy command in your .logout file so that your tickets are destroyed automatically when you logout.

OPTIONS

The following options are supported:

- `-c cache_name`
  Uses cache_name as the credentials (ticket) cache name and location. If this option is not used, the default cache name and location are used.

- `-q`
  Runs quietly. Your terminal will not beep when kdestroy fails to destroy the tickets.

ENVIRONMENT VARIABLES

kdestroy uses the following environment variable:

- `KRB5CCNAME`
  Location of the credentials (ticket) cache.

FILES

- `/tmp/krb5cc_uid`
  Default credentials cache (uid is the decimal UID of the user).

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWkrbu</td>
</tr>
</tbody>
</table>

SEE ALSO

kinit(1), klist(1), attributes(5), SEAM(5)

BUGS

Only the tickets in the specified credentials cache are destroyed. Separate ticket caches are used to hold root instance and password changing tickets. These files should probably be destroyed too, or all of a user’s tickets should be kept in a single credential cache.

AUTHORS

Steve Miller, MIT Project Athena/Digital Equipment Corporation; Clifford Neuman, MIT Project Athena Bill Sommerfeld, MIT Project Athena
keylogin(1)

NAME  keylogin – decrypt and store secret key with keyserv

SYNOPSIS  /usr/bin/keylogin [-r]

DESCRIPTION  The keylogin command prompts for a password, and uses it to decrypt the user's secret key. The key may be found in the /etc/publickey file (see publickey(4)) or the NIS map “publickey.byname” or the NIS+ table “cred.org_dir” in the user's home domain. The sources and their lookup order are specified in the /etc/nsswitch.conf file. See nsswitch.conf(4). Once decrypted, the user's secret key is stored by the local key server process, keyserv(1M). This stored key is used when issuing requests to any secure RPC services, such as NFS or NIS+. The program keylogout(1) can be used to delete the key stored by keyserv.

keylogin will fail if it cannot get the caller's key, or the password given is incorrect. For a new user or host, a new key can be added using newkey(1M), nisaddcred(1M), or nisclient(1M).

If multiple authentication mechanisms are configured for the system, each of the configured mechanism's secret key will be decrypted and stored by keyserv(1M). See nisauthconf(1M) for information on configuring multiple authentication mechanisms.

OPTIONS  -r

Update the /etc/.rootkey file. This file holds the unencrypted secret key of the superuser. Only the superuser may use this option. It is used so that processes running as superuser can issue authenticated requests without requiring that the administrator explicitly run keylogin as superuser at system startup time. See keyserv(1M). The -r option should be used by the administrator when the host's entry in the publickey database has changed, and the /etc/.rootkey file has become out-of-date with respect to the actual key pair stored in the publickey database. The permissions on the /etc/.rootkey file are such that it may be read and written by the superuser but by no other user on the system.

If multiple authentication mechanisms are configured for the system, each of the configured mechanism's secret keys will be stored in the /etc/.rootkey file.

FILES  /etc/.rootkey  superuser's secret key

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO  chkey(1), keylogout(1), login(1), keyserv(1M), newkey(1M), nisaddcred(1M), nisauthconf(1M), nisclient(1M), nsswitch.conf(4), publickey(4), attributes(5)
NIS+ might not be supported in future releases of the Solaris™ Operating Environment. Tools to aid the migration from NIS+ to LDAP are available in the Solaris 9 operating environment. For more information, visit http://www.sun.com/directory/nisplus/transition.html.
keylogout(1)

NAME  keylogout – delete stored secret key with keyserv

SYNOPSIS  /usr/bin/keylogout [-f]

DESCRIPTION  keylogout deletes the key stored by the key server process keyserv(1M). Further access to the key is revoked; however, current session keys may remain valid until they expire or are refreshed.

Deleting the keys stored by keyserv will cause any background jobs or scheduled at(1) jobs that need secure RPC services to fail. Since only one copy of the key is kept on a machine, it is a bad idea to place a call to this command in your .logout file since it will affect other sessions on the same machine.

If multiple NIS+ authentication mechanisms are configured for the system, then all keys stored by the key server process will be deleted, including keys that are no longer configured.

OPTIONS  

- f  Force keylogout to delete the secret key for the superuser. By default, keylogout by the superuser is disallowed because it would break all RPC services, such as NFS, that are started by the superuser.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO  at(1), chkey(1), login(1), keylogin(1), keyserv(1M), newkey(1M), nisauthconf(1M), publickey(4), attributes(5)

NOTES  NIS+ might not be supported in future releases of the Solaris™ Operating Environment. Tools to aid the migration from NIS+ to LDAP are available in the Solaris 9 operating environment. For more information, visit http://www.sun.com/directory/nisplus/transition.html.
kill(1)

NAME
  kill – terminate or signal processes

SYNOPSIS
  /usr/bin/kill -s signal_name pid...
  /usr/bin/kill -l [exit_status]
  /usr/bin/kill [-signal_name] pid...
  /usr/bin/kill [-signal_number] pid...

DESCRIPTION
  The kill utility sends a signal to the process or processes specified by each pid operand.

  For each pid operand, the kill utility will perform actions equivalent to the kill(2) function called with the following arguments:

  1. The value of the pid operand will be used as the pid argument.
  2. The sig argument is the value specified by the -s option, the -signal_name option, or the -signal_number option, or, if none of these options is specified, by SIGTERM.

  The signaled process must belong to the current user unless the user is the super-user.

  See NOTES for descriptions of the shell built-in versions of kill.

OPTIONS
  The following options are supported:

  -l  (The letter ell.) Writes all values of signal_name supported by the implementation, if no operand is given. If an exit_status operand is given and it is a value of the ? shell special parameter and wait corresponding to a process that was terminated by a signal, the signal_name corresponding to the signal that terminated the process will be written. If an exit_status operand is given and it is the unsigned decimal integer value of a signal number, the signal_name corresponding to that signal will be written. Otherwise, the results are unspecified.

  -s signal_name  Specifies the signal to send, using one of the symbolic names defined in the <signal.h> description. Values of signal_name will be recognized in a case-independent fashion, without the SIG prefix. In addition, the symbolic name 0 will be recognized, representing the signal value zero. The corresponding signal will be sent instead of SIGTERM.

  -signal_name  Equivalent to -s signal_name.

  -signal_number  Specifies a non-negative decimal integer, signal_number, representing the signal to be used instead of SIGTERM, as the sig argument in the effective call to kill(2).

OPERANDS
  The following operands are supported:

  pid  One of the following:
1. A decimal integer specifying a process or process group to be signaled. The process or processes selected by positive, negative and zero values of the `pid` operand will be as described for the `kill` function. If process number 0 is specified, all processes in the process group are signaled. If the first `pid` operand is negative, it should be preceded by — to keep it from being interpreted as an option.

2. A job control job ID that identifies a background process group to be signaled. The job control job ID notation is applicable only for invocations of `kill` in the current shell execution environment.

Note: The job control job ID type of `pid` is available only on systems supporting the job control option.

`exit_status` A decimal integer specifying a signal number or the exit status of a process terminated by a signal.

**USAGE**

Process numbers can be found by using `ps(1)`.

The job control job ID notation is not required to work as expected when `kill` is operating in its own utility execution environment. In either of the following examples:

```bash
example$ nohup kill %1 &
example$ system("kill %1");
```

`kill` operates in a different environment and will not share the shell’s understanding of job numbers.

**OUTPUT**

When the `-l` option is not specified, the standard output will not be used.

When the `-l` option is specified, the symbolic name of each signal will be written in the following format:

```
"%s%c", <signal_name>, <separator>
```

where the `<signal_name>` is in upper-case, without the `SIG` prefix, and the `<separator>` will be either a newline character or a space character. For the last signal written, `<separator>` will be a newline character.

When both the `-l` option and `exit_status` operand are specified, the symbolic name of the corresponding signal will be written in the following format:

```
"%s
", <signal_name>
```

**EXAMPLES**

**EXAMPLE 1** Sending the kill signal

Any of the commands:
EXAMPLE 1 Sending the kill signal (Continued)

```
example$ kill -9 100 -165
example$ kill -s kill 100 -165
example$ kill -s KILL 100 -165
```

sends the SIGKILL signal to the process whose process ID is 100 and to all processes whose process group ID is 165, assuming the sending process has permission to send that signal to the specified processes, and that they exist.

EXAMPLE 2 Avoiding ambiguity with an initial negative number

To avoid an ambiguity of an initial negative number argument specifying either a signal number or a process group, the former will always be the case. Therefore, to send the default signal to a process group (for example, 123), an application should use a command similar to one of the following:

```
example$ kill -TERM -123
example$ kill -- -123
```

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of kill: LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS

The following exit values are returned:

- `0` At least one matching process was found for each `pid` operand, and the specified signal was successfully processed for at least one matching process.

- `>0` An error occurred.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

SEE ALSO

csh(1), jobs(1), ksh(1), ps(1), sh(1), shell_builtins(1), wait(1), kill(2), signal(3C), signal(3HEAD), attributes(5), environ(5)

sh

The Bourne shell, sh, has a built-in version of `kill` to provide the functionality of the `kill` command for processes identified with a `jobid`. The sh syntax is:

```
kill [-sig ] [ pid ] [ %job ]...
kil -l
```


csh

The C-shell, csh, also has a built-in `kill` command, whose syntax is:

User Commands 571
kill [-sig] [pid] [%job]...
kill -l

The csh kill built-in sends the TERM (terminate) signal, by default, or the signal specified, to the specified process ID, the job indicated, or the current job. Signals are either given by number or by name. There is no default. Typing kill does not send a signal to the current job. If the signal being sent is TERM (terminate) or HUP (hangup), then the job or process is sent a CONT (continue) signal as well.

-1 Lists the signal names that can be sent.

The syntax of the ksh kill is:

kill [-sig] [pid] [%job]...
kill -l

The ksh kill sends either the TERM (terminate) signal or the specified signal to the specified jobs or processes. Signals are either given by number or by names (as given in signal(3HEAD) stripped of the SIG prefix). If the signal being sent is TERM (terminate) or HUP (hangup), then the job or process will be sent a CONT (continue) signal if it is stopped. The argument job can be the process id of a process that is not a member of one of the active jobs. In the second form, kill -l, the signal numbers and names are listed.
The `kinit` command is used to obtain and cache an initial ticket-granting ticket (credential) for `principal`. This ticket is used for authentication by the Kerberos system. Notice that only users with Kerberos principals can use the Kerberos system. For information about Kerberos principals, see SEAM(5).

When you use `kinit` without options, the utility prompts for your `principal` and Kerberos password, and tries to authenticate your login with the local Kerberos server. The `principal` can be specified on the command line if desired.

If Kerberos authenticates the login attempt, `kinit` retrieves your initial ticket-granting ticket and puts it in the ticket cache. By default your ticket will be stored in the file `/tmp/krb5cc_uid`, where `uid` specifies your user identification number. Tickets expire after a specified lifetime, after which `kinit` must be run again. Any existing contents of the cache are destroyed by `kinit`.

Values specified in the command line override the values specified in the Kerberos configuration file for `lifetime` and `renewable_life`.

The `kdestroy(1)` command may be used to destroy any active tickets before you end your login session.

The following options are supported:

- `-A` Requests address-less tickets.
- `-c cache_name` Uses `cache_name` as the credentials (ticket) cache name and location. If this option is not used, the default cache name and location are used.
- `-f` Requests forwardable tickets.
- `-k [-t keytab_file]` Requests a host ticket, obtained from a key in the local host's `keytab` file. The name and location of the keytab file may be specified with the `-t keytab_file` option; otherwise the default name and location will be used.
- `-l lifetime` Requests a ticket with the lifetime `lifetime`. If the `-l` option is not specified, the default ticket lifetime (configured by each site) is used. Specifying a ticket lifetime longer than the maximum ticket lifetime (configured by each site) results in a ticket with the maximum lifetime. See the Time Formats section for the valid time duration formats that you can specify for...
lifetime. See kdc.conf(4) and kadmin(1M) (for getprinc command to verify the lifetime values for the server principal).

The lifetime of the tickets returned will be the minimum of the following:

- Value specified in the command line.
- Value specified in the KDC configuration file.
- Value specified in the Kerberos data base for the server principal. In the case of kinit, it is krbtgt/realm name.
- Value specified in the Kerberos database for the user principal.

-p Requests proxiable tickets.

-x renewable_life Requests renewable tickets, with a total lifetime of renewable_life. See the Time Formats section for the valid time duration formats that you can specify for renewable_life. See kdc.conf(4) and kadmin(1M) (for getprinc command to verify the lifetime values for the server principal).

The renewable lifetime of the tickets returned will be the minimum of the following:

- Value specified in the command line.
- Value specified in the KDC configuration file.
- Value specified in the Kerberos data base for the server principal. In the case of kinit, it is krbtgt/realm name.
- Value specified in the Kerberos database for the user principal.

-R Requests renewal of the ticket-granting ticket. Notice that an expired ticket cannot be renewed, even if the ticket is still within its renewable life.

-s start_time Requests a postdated ticket, valid starting at start_time. Postdated tickets are issued with the invalid flag set, and need to be fed back to the KDC before use. See the Time Formats section for either the valid absolute time or time duration formats that you can specify for start_time. kinit attempts to match an absolute time first before trying to match a time duration.

-S service_name Specifies an alternate service name to use when getting initial tickets.
Requests that the ticket granting ticket in the cache (with the invalid flag set) be passed to the KDC for validation. If the ticket is within its requested time range, the cache is replaced with the validated ticket.

**Time Formats**

The following absolute time formats can be used for the `-s start_time` option. The examples are based on the date and time of July 2, 1999, 1:35:30 p.m.

<table>
<thead>
<tr>
<th>Absolute Time Format</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ymmdhhhmm[ss]</code></td>
<td>990702133530</td>
</tr>
<tr>
<td><code>hhmm[ss]</code></td>
<td>133530</td>
</tr>
<tr>
<td><code>yy.mm.dd hh:mm:ss</code></td>
<td>99:07:02:13:35:30</td>
</tr>
<tr>
<td><code>hh:mm:ss</code></td>
<td>13:35:30</td>
</tr>
<tr>
<td><code>ldate:time</code></td>
<td>07-07-99:13:35:30</td>
</tr>
<tr>
<td><code>dd-month-yyyyhh:mm:ss</code></td>
<td>02-july-1999:13:35:30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dd</code></td>
<td>day</td>
</tr>
<tr>
<td><code>hh</code></td>
<td>hour (24-hour clock)</td>
</tr>
<tr>
<td><code>mm</code></td>
<td>minutes</td>
</tr>
<tr>
<td><code>ss</code></td>
<td>seconds</td>
</tr>
<tr>
<td><code>yy</code></td>
<td>year within century (0-68 is 2000 to 2068; 69-99 is 1969 to 1999)</td>
</tr>
<tr>
<td><code>yyyy</code></td>
<td>year including century</td>
</tr>
<tr>
<td><code>month</code></td>
<td>locale’s full or abbreviated month name</td>
</tr>
<tr>
<td><code>ldate</code></td>
<td>locale’s appropriate date representation</td>
</tr>
<tr>
<td><code>ltime</code></td>
<td>locale’s appropriate time representation</td>
</tr>
</tbody>
</table>

The following time duration formats can be used for the `-l lifetime`, `-r renewable_life`, and `-s start_time` options. The examples are based on the time duration of 14 days, 7 hours, 5 minutes, and 30 seconds.

<table>
<thead>
<tr>
<th>Time Duration Format</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>#d</code></td>
<td>14d</td>
</tr>
</tbody>
</table>
# Time Duration Format

<table>
<thead>
<tr>
<th>Format</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>#h</td>
<td>7h</td>
</tr>
<tr>
<td>#m</td>
<td>5m</td>
</tr>
<tr>
<td>#s</td>
<td>30s</td>
</tr>
<tr>
<td>#d#h#m#s</td>
<td>14d7h5m30s</td>
</tr>
<tr>
<td>#h#m[#s]</td>
<td>7h5m30s</td>
</tr>
<tr>
<td>days-hh:mm:ss</td>
<td>14-07:05:30</td>
</tr>
<tr>
<td>hours:mm:ss</td>
<td>7:05:30</td>
</tr>
</tbody>
</table>

## Delimiter Description

<table>
<thead>
<tr>
<th>Delimiter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>number of days</td>
</tr>
<tr>
<td>h</td>
<td>number of hours</td>
</tr>
<tr>
<td>m</td>
<td>number of minutes</td>
</tr>
<tr>
<td>s</td>
<td>number of seconds</td>
</tr>
</tbody>
</table>

## Variable Description

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>number</td>
</tr>
<tr>
<td>days</td>
<td>number of days</td>
</tr>
<tr>
<td>hours</td>
<td>number of hours</td>
</tr>
<tr>
<td>hh</td>
<td>hour (24-hour clock)</td>
</tr>
<tr>
<td>mm</td>
<td>minutes</td>
</tr>
<tr>
<td>ss</td>
<td>seconds</td>
</tr>
</tbody>
</table>

## kinit Environment Variables

kinit uses the following environment variable:

- **KRB5CCNAME**: Location of the credentials (ticket) cache.

## kinit Files

- **/tmp/krb5cc_uid**: Default credentials cache (`uid` is the decimal UID of the user).
- **/etc/krb5/krb5.keytab**: Default location for the local host’s keytab file.
- **/etc/krb5/krb5.conf**: Default location for the local host’s configuration file. See `krb5.conf(4)`.

## kinit Attributes

See attributes(5) for descriptions of the following attributes:
<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWkrbu</td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Evolving</td>
</tr>
</tbody>
</table>

**SEE ALSO**

kdestroy(1), klist(1), kadmin(1M), kdc.conf(4), krb5.conf(4), attributes(5), SEAM(5)

**AUTHORS**

Steve Miller, MIT Project Athena/Digital Equipment Corporation; Clifford Neuman, MIT Project Athena
klist(1)

NAME  klist – list currently held Kerberos tickets

SYNOPSIS  /usr/bin/klist [-e] [-a [-n]] [ [-c] [cache_name] [-f] [-s]] [-k
   [-t] [keytab_file] [-K]]

DESCRIPTION  The klist utility prints the name of the credentials cache, the identity of the principal
that the tickets are for (as listed in the ticket file), and the principal names of all
Kerberos tickets currently held by the user, along with the issue and expiration time
for each authenticator. Principal names are listed in the form name/instance@realm,
with the '/' omitted if the instance is not included, and the '@' omitted if the realm is
not included.

If cache_file or keytab_name is not specified, klist will display the credentials in the
default credentials cache or keytab files as appropriate. By default, your ticket will be
stored in the file /tmp/krb5cc_uid, where uid is the current user-ID of the user.

OPTIONS  The following options are supported:

-a
Displays list of addresses in credentials. Uses the
configured nameservice to translate numeric network
addresses to the associated hostname if possible.

-c [cache_name]
Lists tickets held in a credentials cache. This is the
default if neither -c nor -k is specified.

-e
Displays the encryption types of the session key and
the ticket for each credential in the credential cache, or
each key in the keytab file.

-f
Shows the flags present in the credentials, using the
following abbreviations:

F  Forwardable
f  forwarded
P  Proxiable
p  proxy
D  postDateable
d  postdated
R  Renewable
I  Initial
i  invalid

-k [keytab_file]
List keys held in a keytab file.

-K
Displays the value of the encryption key in each keytab
entry in the keytab file.
-n Shows numeric IP addresses instead of reverse-resolving addresses. Only valid with -a option.

-s Causes klist to run silently (produce no output), but to still set the exit status according to whether it finds the credentials cache. The exit status is '0' if klist finds a credentials cache, and '1' if it does not.

-t Displays the time entry timestamps for each keytab entry in the keytab file.

**ENVIRONMENT VARIABLES**

klist uses the following environment variable:

<table>
<thead>
<tr>
<th>ENVIRONMENT VARIABLE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>KRB5CCNAME</td>
<td>Location of the credentials (ticket) cache.</td>
</tr>
</tbody>
</table>

**FILES**

- /tmp/krb5cc_uid (uid is the decimal UID of the user) Default credentials cache. 
- /etc/krb5/krb5.keytab Default location for the local host’s keytab file. 
- /etc/krb5/krb5.conf Default location for the local host’s configuration file. See krb5.conf(4).

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWkrbu</td>
</tr>
<tr>
<td>Interface Stability</td>
<td></td>
</tr>
<tr>
<td>Command arguments</td>
<td>Evolving</td>
</tr>
<tr>
<td>Command output</td>
<td>Unstable</td>
</tr>
</tbody>
</table>

**SEE ALSO**
kdestroy(1), kinit(1), krb5.conf(4), attributes(5), SEAM(5)

**BUGS**

When reading a file as a service key file, very little error checking is performed.
The kpasswd command is used to change a Kerberos principal’s password. kpasswd prompts for the current Kerberos password, which is used to obtain a changepw ticket from the KDC for the user’s Kerberos realm. If kpasswd successfully obtains the changepw ticket, the user is prompted twice for the new password, and the password is changed.

If the principal is governed by a policy that specifies the length and/or number of character classes required in the new password, the new password must conform to the policy. (The five character classes are lower case, upper case, numbers, punctuation, and all other characters.)

The following operand is supported:

```
principal
```

Change the password for the Kerberos principal principal. Otherwise, the principal is derived from the identity of the user invoking the kpasswd command.

```
/tmp/ovsec_adm.xxxxx
```

Temporary credentials cache for the lifetime of the password changing operation. (xxxxxx is a random string.)

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWkrbu</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

If kpasswd is suspended, the changepw tickets may not be destroyed.
NAME
ksh, rksh – KornShell, a standard/restricted command and programming language

SYNOPSIS
/usr/bin/ksh [± abCefhikmnoprstuvx] [± o option...] [arg...]
/usr/bin/ksh -c [± abCefhikmnoprstuvx] [± o option...] command_string
[command_name [arg...]]
/usr/xpg4/bin/sh [± abCefhikmnoprstuvx] [± o option...][arg...]
/usr/xpg4/bin/sh -c [± abCefhikmnoprstuvx] [± o option...][command_string
[command_name [arg...]]]
/usr/bin/rksh [± abCefhikmnoprstuvx] [± o option...][arg...]
/usr/bin/rksh -c [± abCefhikmnoprstuvx] [± o option...][command_string
[command_name [arg...]]]

DESCRIPTION
The /usr/xpg4/bin/sh utility is a standards compliant shell. This utility provides all the functionality of /usr/bin/ksh, except in cases discussed below where differences in behavior exist. See Arithmetic Expansions section for details.

The /usr/bin/ksh is a command and programming language that executes commands read from a terminal or a file. rksh is a restricted version of the command interpreter ksh; it is used to set up login names and execution environments whose capabilities are more controlled than those of the standard shell. See Invocation below for the meaning of arguments to the shell.

Definitions
A metacharacter is one of the following characters:

; & ( ) | < > NEWLINE SPACE TAB

A blank is a TAB or a SPACE. An identifier is a sequence of letters, digits, or underscores starting with a letter or underscore. Identifiers are used as names for functions and variables. A word is a sequence of characters separated by one or more non-quoted metacharacters.

A command is a sequence of characters in the syntax of the shell language. The shell reads each command and carries out the desired action either directly or by invoking separate utilities. A special-command is a command that is carried out by the shell without creating a separate process. Except for documented side effects, most special commands can be implemented as separate utilities.

Commands
A simple-command is a sequence of blank-separated words which may be preceded by a variable assignment list. (See Environment below.) The first word specifies the name of the command to be executed. Except as specified below, the remaining words are passed as arguments to the invoked command. The command name is passed as argument 0 (see exec(2)). The value of a simple-command is its exit status if it terminates normally. If it terminates abnormally due to receipt of a signal, the value is the signal number plus 128. See signal(3HEAD) for a list of signal values. Obviously, normal exit status values 129 to 255 cannot be distinguished from abnormal exit caused by receiving signal numbers 1 to 127.
A pipeline is a sequence of one or more commands separated by |. The standard output of each command but the last is connected by a pipe(2) to the standard input of the next command. Each command is run as a separate process; the shell waits for the last command to terminate. The exit status of a pipeline is the exit status of the last command.

A list is a sequence of one or more pipelines separated by ;, &, ||, or |, and optionally terminated by ;, & or | &. Of these five symbols, ;, & and || have equal precedence, which is lower than that of & & and | |. The symbols & & and | | also have equal precedence. A semicolon (;) causes sequential execution of the preceding pipeline; an ampersand (&) causes asynchronous execution of the preceding pipeline (that is, the shell does not wait for that pipeline to finish). The symbol | & causes asynchronous execution of the preceding command or pipeline with a two-way pipe established to the parent shell.

The standard input and output of the spawned command can be written to and read from by the parent shell using the -p option of the special commands read and print described in Special Commands. The symbol && (||) causes the list following it to be executed only if the preceding pipeline returns 0 (or a non-zero) value. An arbitrary number of new-lines may appear in a list, instead of a semicolon, to delimit a command.

A command is either a simple-command or one of the following. Unless otherwise stated, the value returned by a command is that of the last simple-command executed in the command.

```
for identifier [ in word ... ] ; do list ; done
```

Each time a for command is executed, identifier is set to the next word taken from the in word list. If in word ... is omitted, then the for command executes the do list once for each positional parameter that is set (see Parameter Substitution below). Execution ends when there are no more words in the list.

```
select identifier [ in word ... ] ; do list ; done
```

A select command prints to standard error (file descriptor 2), the set of words, each preceded by a number. If in word ... is omitted, then the positional parameters are used instead (see Parameter Substitution below). The PS3 prompt is printed and a line is read from the standard input. If this line consists of the number of one of the listed words, then the value of the variable identifier is set to the word corresponding to this number. If this line is empty the selection list is printed again. Otherwise the value of the variable identifier is set to NULL. (See Blank Interpretation about NULL). The contents of the line read from standard input is saved in the shell variable REPLY. The list is executed for each selection until a break or EOF is encountered. If the REPLY variable is set to NULL by the execution of list, then the selection list is printed before displaying the PS3 prompt for the next selection.

```
case word in [ pattern [ | pattern ] ] list ; ] ... esac
```

A case command executes the list associated with the first pattern that matches word. The form of the patterns is the same as that used for file-name generation (see File Name Generation below).
if list ; then list ; [ elif list ; then list ; ... ] [ else list ; ] fi

The list following if is executed and, if it returns an exit status of 0, the list following the first then is executed. Otherwise, the list following elif is executed and, if its value is 0, the list following the next then is executed. Failing that, the else list is executed. If no else list or then list is executed, then the if command returns 0 exit status.

while list ; do list ; done
until list ; do list ; done

A while command repeatedly executes the while list and, if the exit status of the last command in the list is 0, executes the do list; otherwise the loop terminates. If no commands in the do list are executed, then the while command returns 0 exit status. until may be used in place of while to negate the loop termination test.

(list)

Execute list in a separate environment. Notice that if two adjacent open parentheses are needed for nesting, a space must be inserted to avoid arithmetic evaluation as described below.

{list}

list is simply executed. Notice that, unlike the metacharacters ( and ), { and } are reserved words and must occur at the beginning of a line or after a ; in order to be recognized.

[ [expression] ]

Evaluates expression and returns 0 exit status when expression is true. See Conditional Expressions below, for a description of expression.

function identifier { list ; }

identifier( ) { list ; }

Define a function which is referenced by identifier. The body of the function is the list of commands between { and }. (See Functions below).

time pipeline

The pipeline is executed and the elapsed time as well as the user and system time are printed to standard error.

The following reserved words are only recognized as the first word of a command and when not quoted:

! if then else elif fi case esac for while until do done [

function select time [ [ ] ]

Comments

A word beginning with # causes that word and all the following characters up to a new-line to be ignored.

Aliasing

The first word of each command is replaced by the text of an alias if an alias for this word has been defined. An alias name consists of any number of characters excluding metacharacters, quoting characters, file expansion characters, parameter and command substitution characters, and =. The replacement string can contain any valid shell script including the metacharacters listed above. The first word of each command

User Commands  583
in the replaced text, other than any that are in the process of being replaced, will be
tested for aliases. If the last character of the alias value is a blank then the word
following the alias will also be checked for alias substitution. Aliases can be used to
redefine special builtin commands but cannot be used to redefine the reserved words
listed above. Aliases can be created, listed, and exported with the alias command
and can be removed with the unalias command. Exported aliases remain in effect
for scripts invoked by name, but must be reinitialized for separate invocations of the
shell (see Invocation below). To prevent infinite loops in recursive aliasing, if the
shell is not currently processing an alias of the same name, the word will be replaced
by the value of the alias; otherwise, it will not be replaced.

Aliasing is performed when scripts are read, not while they are executed. Therefore,
for an alias to take effect, the alias definition command has to be executed before the
command which references the alias is read.

Aliases are frequently used as a short hand for full path names. An option to the
aliasing facility allows the value of the alias to be automatically set to the full
pathname of the corresponding command. These aliases are called tracked aliases. The
value of a tracked alias is defined the first time the corresponding command is looked
up and becomes undefined each time the PATH variable is reset. These aliases remain
tracked so that the next subsequent reference will redefine the value. Several tracked
aliases are compiled into the shell. The -h option of the set command makes each
referenced command name into a tracked alias.

The following exported aliases are compiled into (and built-in to) the shell but can be
unset or redefined:

```
autoload='typeset -fu'
false='let 0'
functions='typeset -f'
hash='alias -t'
history='fc -l'
integer='typeset -i'
nohup='nohup '
r='fc -e -r'
true=';'
type='whence --v'
```

An example concerning trailing blank characters and reserved words follows. If the
user types:

```
$ alias foo="/bin/ls 
$ alias while="/"
```

the effect of executing:

```
$ while true
  do
    echo "Hello, World"
  done
```

is a never-ending sequence of Hello, World strings to the screen. However, if the
user types:

```
$ while true
  do
    echo "Hello, World"
  done
```
foo while

the result will be an ls listing of /. Since the alias substitution for foo ends in a space character, the next word is checked for alias substitution. The next word, while, has also been aliased, so it is substituted as well. Since it is not in the proper position as a command word, it is not recognized as a reserved word.

If the user types:

$ foo; while

while retains its normal reserved-word properties.

Tilde Substitution

After alias substitution is performed, each word is checked to see if it begins with an unquoted ~. If it does, then the word up to a / is checked to see if it matches a user name. If a match is found, the ~ and the matched login name are replaced by the login directory of the matched user. This is called a tilde substitution. If no match is found, the original text is left unchanged. A ~ by itself, or in front of a /, is replaced by $HOME. A ~ followed by a + or − is replaced by $PWD and $OLDPWD, respectively.

In addition, tilde substitution is attempted when the value of a variable assignment begins with a ~.

Tilde Expansion

A tilde-prefix consists of an unquoted tilde character at the beginning of a word, followed by all of the characters preceding the first unquoted slash in the word, or all the characters in the word if there is no slash. In an assignment, multiple tilde-prefixes can be used: at the beginning of the word (that is, following the equal sign of the assignment), following any unquoted colon or both. A tilde-prefix in an assignment is terminated by the first unquoted colon or slash. If none of the characters in the tilde-prefix are quoted, the characters in the tilde-prefix following the tilde are treated as a possible login name from the user database.

A portable login name cannot contain characters outside the set given in the description of the LOGNAME environment variable. If the login name is null (that is, the tilde-prefix contains only the tilde), the tilde-prefix will be replaced by the value of the variable HOME. If HOME is unset, the results are unspecified. Otherwise, the tilde-prefix will be replaced by a pathname of the home directory associated with the login name obtained using the getpwnam function. If the system does not recognize the login name, the results are undefined.

Tilde expansion generally occurs only at the beginning of words, but an exception based on historical practice has been included:

PATH=/posix/bin:~/dgk/bin

is eligible for tilde expansion because tilde follows a colon and none of the relevant characters is quoted. Consideration was given to prohibiting this behavior because any of the following are reasonable substitutes:

PATH=$(printf %s ~karels/bin :~bostic/bin)

for Dir in ~maart/bin ~srb/bin .
do
With the first command, explicit colons are used for each directory. In all cases, the shell performs tilde expansion on each directory because all are separate words to the shell.

Notice that expressions in operands such as:

```
make -k mumble LIBDIR=~chet/lib
```

do not qualify as shell variable assignments and tilde expansion is not performed (unless the command does so itself, which `make` does not).

The special sequence `$~` has been designated for future implementations to evaluate as a means of forcing tilde expansion in any word.

Because of the requirement that the word not be quoted, the following are not equivalent; only the last will cause tilde expansion:

```
~hlj/ ~h\lj/ ~"hlj"/ ~hlj/ ~hlj/
```

The results of giving tilde with an unknown login name are undefined because the KornShell `--` and `--` constructs make use of this condition, but, in general it is an error to give an incorrect login name with tilde. The results of having `HOME` unset are unspecified because some historical shells treat this as an error.

The standard output from a `command` enclosed in parenthesis preceded by a dollar sign (that is, `$ (command)` or a pair of grave accents (`'`) may be used as part or all of a word; trailing new-lines are removed. In the second (archaic) form, the string between the quotes is processed for special quoting characters before the command is executed. (See Quoting below.) The command substitution `$ (cat file)` can be replaced by the equivalent but faster `$ <file)`. Command substitution of most special commands that do not perform input/output redirection are carried out without creating a separate process.

Command substitution allows the output of a command to be substituted in place of the command name itself. Command substitution occurs when the command is enclosed as follows:

```
$(command)
```

or (backquoted version):

```
`command`
```

The shell will expand the command substitution by executing `command` in a subshell environment and replacing the command substitution (the text of `command` plus the enclosing `$(` or backquotes) with the standard output of the command, removing sequences of one or more newline characters at the end of the substitution. Embedded
newline characters before the end of the output will not be removed; however, they may be treated as field delimiters and eliminated during field splitting, depending on the value of IFS and quoting that is in effect.

Within the backquoted style of command substitution, backslash shall retain its literal meaning, except when followed by:

```
$  `  \
```

(dollar-sign, backquote, backslash). The search for the matching backquote is satisfied by the first backquote found without a preceding backslash. During this search, if a non-escaped backquote is encountered within a shell comment, a here-document, an embedded command substitution of the $(command) form, or a quoted string, undefined results occur. A single- or double-quoted string that begins, but does not end, within the `' . . .'` sequence produces undefined results.

With the $(command) form, all characters following the open parenthesis to the matching closing parenthesis constitute the command. Any valid shell script can be used for command, except:

- A script consisting solely of redirections produces unspecified results.
- See the restriction on single subshells described below.

The results of command substitution will not be field splitting and pathname expansion processed for further tilde expansion, parameter expansion, command substitution or arithmetic expansion. If a command substitution occurs inside double-quotes, it will not be performed on the results of the substitution.

Command substitution can be nested. To specify nesting within the backquoted version, the application must precede the inner backquotes with backslashes; for example:

```
`\ `command `\`
```

The $() form of command substitution solves a problem of inconsistent behavior when using backquotes. For example:

<table>
<thead>
<tr>
<th>Command</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\\$x</code></td>
<td>$x</td>
</tr>
<tr>
<td><code>\</code>echo <code>\\$x</code>&quot;&quot;`</td>
<td>$x</td>
</tr>
<tr>
<td><code>echo $(\</code>\$x&quot;)`</td>
<td>$x</td>
</tr>
</tbody>
</table>

Additionally, the backquoted syntax has historical restrictions on the contents of the embedded command. While the new $() form can process any kind of valid embedded script, the backquoted form cannot handle some valid scripts that include backquotes. For example, these otherwise valid embedded scripts do not work in the left column, but do work on the right:
Because of these inconsistent behaviors, the backquoted variety of command substitution is not recommended for new applications that nest command substitutions or attempt to embed complex scripts.

If the command substitution consists of a single subshell, such as:

```
$( (command ) )
```

a portable application must separate the $ ( and ) into two tokens (that is, separate them with white space). This is required to avoid any ambiguities with arithmetic expansion.

Arithmetic Expansion

An arithmetic expression enclosed in double parentheses preceded by a dollar sign ($ ( (arithmetic-expression) ) ) is replaced by the value of the arithmetic expression within the double parenthesis. Arithmetic expansion provides a mechanism for evaluating an arithmetic expression and substituting its value. The format for arithmetic expansion is as follows:

```
$((expression))
```

The expression is treated as if it were in double-quotes, except that a double-quote inside the expression is not treated specially. The shell will expand all tokens in the expression for parameter expansion, command substitution and quote removal.

Next, the shell will treat this as an arithmetic expression and substitute the value of the expression. The arithmetic expression will be processed according to the rules of the ISO C with the following exceptions:

- Only integer arithmetic is required.
The sizeof() operator and the prefix and postfix ++ and −− operators are not required.

Selection, iteration, and jump statements are not supported.

/usr/bin/ksh and /usr/bin/rksh treat prefix 0 through 9 as decimal constants. See the examples below.

<table>
<thead>
<tr>
<th>Command</th>
<th>Result in /bin/ksh</th>
<th>Result in /usr/xpg4/bin/sh</th>
</tr>
</thead>
<tbody>
<tr>
<td>echo $(010+10)</td>
<td>20</td>
<td>18</td>
</tr>
<tr>
<td>echo $(010+10)</td>
<td>29</td>
<td>error</td>
</tr>
<tr>
<td>[ 10 −le $(011)]</td>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>

As an extension, the shell may recognize arithmetic expressions beyond those listed. If the expression is invalid, the expansion will fail and the shell will write a message to standard error indicating the failure.

A simple example using arithmetic expansion:

```bash
# repeat a command 100 times
x=100
while [ $x -gt 0 ]
do
command
  x=$((x-1))
done
```

This feature is available in SunOS and only on versions of the UNIX operating system that support the `/dev/fd` directory for naming open files. Each command argument of the form <(list) or >(list) will run process list asynchronously connected to some file in /dev/fd. The name of this file will become the argument to the command. If the form with > is selected, then writing on this file will provide input for list. If < is used, then the file passed as an argument will contain the output of the list process. For example,

```
paste <(cut -f1 file1) <(cut -f3 file2) | tee >(process1) >(process2)
```

cuts fields 1 and 3 from the files file1 and file2, respectively, pastes the results together, and sends it to the processes process1 and process2, as well as putting it onto the standard output. Note that the file, which is passed as an argument to the command, is a UNIX pipe(2) so programs that expect to lseek(2) on the file will not work.

A parameter is an identifier, one or more digits, or any of the characters *, @, #, ?, ~, $, and !. A variable (a parameter denoted by an identifier) has a value and zero or more attributes. Variables can be assigned values and attributes by using the typeset special command. The attributes supported by the shell are described later with the typeset special command. Exported variables pass values and attributes to the environment.
The shell supports a one-dimensional array facility. An element of an array variable is referenced by a subscript. A subscript is denoted by a `[`, followed by an arithmetic expression (see Arithmetic Evaluation below) followed by a `]`. To assign values to an array, use `set -A name value ...`. The value of all subscripts must be in the range of 0 through 4095. Arrays need not be declared. Any reference to a variable with a valid subscript is legal and an array will be created if necessary. Referencing an array without a subscript is equivalent to referencing the element 0. If an array identifier with subscript `*` or `@` is used, then the value for each of the elements is substituted (separated by a field separator character).

The value of a variable may be assigned by writing:

```
name=value [ name=value ] ...
```

If the integer attribute, `-i`, is set for name, the value is subject to arithmetic evaluation as described below.

Positional parameters, parameters denoted by a number, may be assigned values with the `set` special command. Parameter `$0` is set from argument zero when the shell is invoked. If `parameter` is one or more digits then it is a positional parameter. A positional parameter of more than one digit must be enclosed in braces.

The format for parameter expansion is as follows:

```
${expression}
```

where `expression` consists of all characters until the matching `}`. Any `}` escaped by a backslash or within a quoted string, and characters in embedded arithmetic expansions, command substitutions and variable expansions, are not examined in determining the matching `}`.

The simplest form for parameter expansion is:

```
${parameter}
```

The value, if any, of `parameter` will be substituted.

The parameter name or symbol can be enclosed in braces, which are optional except for positional parameters with more than one digit or when `parameter` is followed by a character that could be interpreted as part of the name. The matching closing brace will be determined by counting brace levels, skipping over enclosed quoted strings and command substitutions.

If the parameter name or symbol is not enclosed in braces, the expansion will use the longest valid name whether or not the symbol represented by that name exists. When the shell is scanning its input to determine the boundaries of a name, it is not bound by its knowledge of what names are already defined. For example, if `F` is a defined shell variable, the command:
echo $Fred

does not echo the value of $F followed by red; it selects the longest possible valid
name, Fred, which in this case might be unset.

If a parameter expansion occurs inside double-quotes:

- Pathname expansion will not be performed on the results of the expansion.
- Field splitting will not be performed on the results of the expansion, with the
  exception of @.

In addition, a parameter expansion can be modified by using one of the following
formats. In each case that a value of word is needed (based on the state of parameter, as
described below), word will be subjected to tilde expansion, parameter expansion,
command substitution and arithmetic expansion. If word is not needed, it will not be
expanded. The } character that delimits the following parameter expansion
modifications is determined as described previously in this section and in dquote.
(For example, ${foo-bar}xyz} would result in the expansion of foo followed by
the string xyz} if foo is set, else the string barxyz}).

$\{parameter:−word\} Use Default Values. If parameter is unset or null,
the expansion of word will be substituted; otherwise, the value of parameter will be substituted.

$\{parameter:=word\} Assign Default Values. If parameter is unset or
null, the expansion of word will be assigned to parameter. In all cases, the final value of parameter will
be substituted. Only variables, not positional
parameters or special parameters, can be assigned in this way.

$\{parameter:?[word]\} Indicate Error if Null or Unset. If parameter
is unset or null, the expansion of word (or a message
indicating it is unset if word is omitted) will be written
to standard error and the shell will exit with a non-zero
exit status. Otherwise, the value of parameter will be
substituted. An interactive shell need not exit.

$\{parameter:+[word]\} Use Alternative Value. If parameter is unset or
null, null will be substituted. Otherwise, the
expansion of word will be substituted.

In the parameter expansions shown previously, use of the colon in the format results
in a test for a parameter that is unset or null. Omission of the colon results in a test
for a parameter that is only unset. The following table summarizes the effect of the
colon:
In all cases shown with “substitute”, the expression is replaced with the value shown. In all cases shown with “assign” parameter is assigned that value, which also replaces the expression.

In all cases shown with “assign” parameter is assigned that value, which also replaces the expression.

The following four varieties of parameter expansion provide for substring processing. In each case, pattern matching notation (see \texttt{patmat}), rather than regular expression notation, will be used to evaluate the patterns. If parameter is \texttt{*} or \texttt{@}, then all the positional parameters, starting with \$1, are substituted (separated by a field separator character).

Enclosing the full parameter expansion string in double-quotes will not cause the following four varieties of pattern characters to be quoted, whereas quoting characters within the braces will have this effect.

\$\{\#\texttt{parameter}\} \quad \texttt{String Length. The length in characters of the value of parameter. If parameter is \texttt{*} or \texttt{@}, then all the positional parameters, starting with \$1, are substituted (separated by a field separator character).}

\$\{\texttt{parameter}\%\texttt{word}\} \quad \texttt{Remove Smallest Suffix Pattern. The word will be expanded to produce a pattern. The parameter expansion then will result in parameter, with the smallest portion of the suffix matched by the pattern deleted.}

\$\{\texttt{parameter}\%\%\texttt{word}\} \quad \texttt{Remove Largest Suffix Pattern. The word will be expanded to produce a pattern. The parameter expansion then will result in parameter, with the largest portion of the suffix matched by the pattern deleted.}
Remove Smallest Prefix Pattern. The word will be expanded to produce a pattern. The parameter expansion then will result in parameter, with the smallest portion of the prefix matched by the pattern deleted.

Remove Largest Prefix Pattern. The word will be expanded to produce a pattern. The parameter expansion then will result in parameter, with the largest portion of the prefix matched by the pattern deleted.

Examples:

\$®word

In this example, \texttt{ls} is executed only if \texttt{x} is null or unset. (The \texttt{($ls)} command substitution notation is explained in Command Substitution above.)

\$®word

unset X
echo \$®abc
abc

\$®word

unset posix
echo \$®
sh: posix: parameter null or not set

\$®word

set a b c
echo \$®posix
posix

\$®word

HOME=/usr/posix
echo \$®HOME
10

\$®word

x=file.c
echo \$®.o
file.o

\$®word

x=posix/src/std
echo \$®*
posix
The following parameters are automatically set by the shell:

# The number of positional parameters in decimal.
- Flags supplied to the shell on invocation or by the `set` command.
? The decimal value returned by the last executed command.
$ The process number of this shell.
- Initially, the value of _ is an absolute pathname of the shell or script being executed as passed in the `environment`. Subsequently it is assigned the last argument of the previous command. This parameter is not set for commands which are asynchronous. This parameter is also used to hold the name of the matching `MAIL` file when checking for mail.
! The process number of the last background command invoked.

ERRNO The value of `errno` as set by the most recently failed system call. This value is system dependent and is intended for debugging purposes.

LINENO The line number of the current line within the script or function being executed.

OLDPWD The previous working directory set by the `cd` command.

OPTARG The value of the last option argument processed by the `getopts` special command.

OPTIND The index of the last option argument processed by the `getopts` special command.

PPID The process number of the parent of the shell.

PWD The present working directory set by the `cd` command.

RANDOM Each time this variable is referenced, a random integer, uniformly distributed between 0 and 32767, is generated. The sequence of random numbers can be initialized by assigning a numeric value to RANDOM.

REPLY This variable is set by the `select` statement and by the `read` special command when no arguments are supplied.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SECONDS</td>
<td>Each time this variable is referenced, the number of seconds since shell invocation is returned. If this variable is assigned a value, then the value returned upon reference will be the value that was assigned plus the number of seconds since the assignment.</td>
</tr>
<tr>
<td>CDPATH</td>
<td>The search path for the cd command.</td>
</tr>
<tr>
<td>COLUMNS</td>
<td>If this variable is set, the value is used to define the width of the edit window for the shell edit modes and for printing select lists.</td>
</tr>
<tr>
<td>EDITOR</td>
<td>If the value of this variable ends in emacs, gmacs, or vi and the VISUAL variable is not set, then the corresponding option (see the set special command below) will be turned on.</td>
</tr>
<tr>
<td>ENV</td>
<td>This variable, when the shell is invoked, is subjected to parameter expansion by the shell and the resulting value is used as a pathname of a file containing shell commands to execute in the current environment. The file need not be executable. If the expanded value of ENV is not an absolute pathname, the results are unspecified. ENV will be ignored if the user's real and effective user IDs or real and effective group IDs are different. This variable can be used to set aliases and other items local to the invocation of a shell. The file referred to by ENV differs from $HOME/.profile in that .profile is typically executed at session startup, whereas the ENV file is executed at the beginning of each shell invocation. The ENV value is interpreted in a manner similar to a dot script, in that the commands are executed in the current environment and the file needs to be readable, but not executable. However, unlike dot scripts, no PATH searching is performed. This is used as a guard against Trojan Horse security breaches.</td>
</tr>
<tr>
<td>FCEDIT</td>
<td>The default editor name for the fc command.</td>
</tr>
<tr>
<td>FPATH</td>
<td>The search path for function definitions. By default, the FPATH directories are searched after the PATH variable. If an executable file is found, then it is read and executed in the current environment. FPATH is searched before PATH when a function with the -u attribute is referenced. The preset alias autoload causes a function with the -u attribute to be created.</td>
</tr>
<tr>
<td>IFS</td>
<td>Internal field separators, normally space, tab, and new-line that are used to separate command words which result from command or parameter substitution and for separating words with the special command read. The first character of the IFS variable is used to separate arguments for the $* substitution (See Quoting below).</td>
</tr>
</tbody>
</table>
### ksh(1)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HISTFILE</strong></td>
<td>If this variable is set when the shell is invoked, then the value is the path name of the file that will be used to store the command history. (See Command re-entry below.)</td>
</tr>
<tr>
<td><strong>HISTSIZE</strong></td>
<td>If this variable is set when the shell is invoked, then the number of previously entered commands that are accessible by this shell will be greater than or equal to this number. The default is 128.</td>
</tr>
<tr>
<td><strong>HOME</strong></td>
<td>The default argument (home directory) for the <code>cd</code> command.</td>
</tr>
<tr>
<td><strong>LC_ALL</strong></td>
<td>This variable provides a default value for the <code>LC_*</code> variables.</td>
</tr>
<tr>
<td><strong>LC_COLLATE</strong></td>
<td>This variable determines the behavior of range expressions, equivalence classes and multi-byte character collating elements within pattern matching.</td>
</tr>
<tr>
<td><strong>LC_CTYPE</strong></td>
<td>Determines how the shell handles characters. When <code>LC_CTYPE</code> is set to a valid value, the shell can display and handle text and filenames containing valid characters for that locale. If <code>LC_CTYPE</code> (see <code>environ(5)</code>) is not set in the environment, the operational behavior of the shell is determined by the value of the <code>LANG</code> environment variable. If <code>LC_ALL</code> is set, its contents are used to override both the <code>LANG</code> and the other <code>LC_*</code> variables.</td>
</tr>
<tr>
<td><strong>LC_MESSAGES</strong></td>
<td>This variable determines the language in which messages should be written.</td>
</tr>
<tr>
<td><strong>LANG</strong></td>
<td>Provide a default value for the internationalization variables that are unset or null. If any of the internationalization variables contains an invalid setting, the utility will behave as if none of the variables had been defined.</td>
</tr>
<tr>
<td><strong>LINENO</strong></td>
<td>This variable is set by the shell to a decimal number representing the current sequential line number (numbered starting with 1) within a script or function before it executes each command. If the user unsets or resets <code>LINENO</code>, the variable may lose its special meaning for the life of the shell. If the shell is not currently executing a script or function, the value of <code>LINENO</code> is unspecified.</td>
</tr>
<tr>
<td><strong>LINES</strong></td>
<td>If this variable is set, the value is used to determine the column length for printing select lists. Select lists will print vertically until about two-thirds of <code>LINES</code> lines are filled.</td>
</tr>
<tr>
<td><strong>MAIL</strong></td>
<td>If this variable is set to the name of a mail file and the <code>MAILPATH</code> variable is not set, then the shell informs the user of arrival of mail in the specified file.</td>
</tr>
<tr>
<td><strong>MAILCHECK</strong></td>
<td>This variable specifies how often (in seconds) the shell will check for changes in the modification time of any of the files specified by the <code>MAILPATH</code> or <code>MAIL</code> variables. The default value is 600 seconds. When the time has elapsed the shell will check before issuing the next prompt.</td>
</tr>
</tbody>
</table>
### MAILPATH
A colon (:) separated list of file names. If this variable is set, then the shell informs the user of any modifications to the specified files that have occurred within the last MAILCHECK seconds. Each file name can be followed by a ? and a message that will be printed. The message will undergo parameter substitution with the variable $_ defined as the name of the file that has changed. The default message is you have mail in $_.

### NLSPATH
Determine the location of message catalogues for the processing of LC_MESSAGES.

### PATH
The search path for commands (see Execution below). The user may not change PATH if executing under rksh (except in .profile).

### PPID
This variable is set by the shell to the decimal process ID of the process that invoked the shell. In a subshell, PPID will be set to the same value as that of the parent of the current shell. For example, echo $PPID and (echo $PPID) would produce the same value.

### PS1
The value of this variable is expanded for parameter substitution to define the primary prompt string which by default is “$ “. The character ! in the primary prompt string is replaced by the command number (see Command Re-entry below). Two successive occurrences of ! will produce a single ! when the prompt string is printed.

### PS2
Secondary prompt string, by default “> “.

### PS3
Selection prompt string used within a select loop, by default “#? “.

### PS4
The value of this variable is expanded for parameter substitution and precedes each line of an execution trace. If omitted, the execution trace prompt is “+ “.

### SHELL
The pathename of the shell is kept in the environment. At invocation, if the basename of this variable is rsh, rksh, or krsh, then the shell becomes restricted.

### TMOUT
If set to a value greater than zero, the shell will terminate if a command is not entered within the prescribed number of seconds after issuing the PS1 prompt. (Notice that the shell can be compiled with a maximum bound for this value which cannot be exceeded.)

### VISUAL
If the value of this variable ends in emacs, gemacs, or vi, then the corresponding option (see Special Command set below) will be turned on.
The shell gives default values to PATH, PS1, PS2, PS3, PS4, MAILCHECK, FCEDIT, TMOUT, and IFS, while HOME, SHELL, ENV, and MAIL are not set at all by the shell (although HOME is set by login(1)). On some systems MAIL and SHELL are also set by login.

After parameter and command substitution, the results of substitutions are scanned for the field separator characters (those found in IFS) and split into distinct arguments where such characters are found. Explicit null arguments ("" or ‘’) are retained. Implicit null arguments (those resulting from parameters that have no values) are removed.

Following substitution, each command word is scanned for the characters *, ?, and [ unless the -f option has been set. If one of these characters appears, the word is regarded as a pattern. The word is replaced with lexicographically sorted file names that match the pattern. If no file name is found that matches the pattern, the word is left unchanged. When a pattern is used for file name generation, the character period (.) at the start of a file name or immediately following a /, as well as the character / itself, must be matched explicitly. A file name beginning with a period will not be matched with a pattern with the period inside parentheses. That is, ls .@(.r*) would locate a file named .restore, but ls @(r*) would not. In other instances of pattern matching, the / and . are not treated specially.

*      Matches any string, including the null string.
?      Matches any single character.
[...]  Matches any one of the enclosed characters. A pair of characters separated by – matches any character lexically between the pair, inclusive. If the first character following the opening "[ " is a "! ", then any character not enclosed is matched. A – can be included in the character set by putting it as the first or last character.

A pattern-list is a list of one or more patterns separated from each other with a |. Composite patterns can be formed with one or more of the following:

? (pattern-list)    Optionally matches any one of the given patterns.
* (pattern-list)    Matches zero or more occurrences of the given patterns.
+ (pattern-list)    Matches one or more occurrences of the given patterns.
@ (pattern-list)    Matches exactly one of the given patterns.
! (pattern-list)    Matches anything, except one of the given patterns.

Quoting

Each of the metacharacters listed above (see Definitions) has a special meaning to the shell and causes termination of a word unless quoted. A character may be quoted (that is, made to stand for itself) by preceding it with a \\. The pair \ NEWLINE is removed. All characters enclosed between a pair of single quote marks (‘’) are quoted. A single quote cannot appear within single quotes. Inside double quote marks
(*"_*), parameter and command substitution occur and \ quotes the characters \ , ' ', " , and $. The meaning of *$ and @$ is identical when not quoted or when used as a parameter assignment value or as a file name. However, when used as a command argument, *$ is equivalent to `$1 $2 $3 ..` , where $d is the first character of the IFS variable, whereas @$ is equivalent to $1 $2 ..$. Inside grave quote marks (` *`), \ quotes the characters \ , ' ', and $. If the grave quotes occur within double quotes, then \ also quotes the character ".

The special meaning of reserved words or aliases can be removed by quoting any character of the reserved word. The recognition of function names or special command names listed below cannot be altered by quoting them.

**Arithmetic Evaluation**

An ability to perform integer arithmetic is provided with the special command `let`. Evaluations are performed using long arithmetic. Constants are of the form [ base# ] n where base is a decimal number between two and thirty-six representing the arithmetic base and n is a number in that base. If base is omitted then base 10 is used.

An arithmetic expression uses the same syntax, precedence, and associativity of expression as the C language. All the integral operators, other than ++, --, ?:, and , are supported. Variables can be referenced by name within an arithmetic expression without using the parameter substitution syntax. When a variable is referenced, its value is evaluated as an arithmetic expression.

An internal integer representation of a variable can be specified with the `-i` option of the `typeset` special command. Arithmetic evaluation is performed on the value of each assignment to a variable with the `-i` attribute. If you do not specify an arithmetic base, the first assignment to the variable determines the arithmetic base. This base is used when parameter substitution occurs.

Since many of the arithmetic operators require quoting, an alternative form of the `let` command is provided. For any command which begins with a `((`, all the characters until a matching `))` are treated as a quoted expression. More precisely, `((..))` is equivalent to `let ".."`.

**Prompting**

When used interactively, the shell prompts with the parameter expanded value of `PS1` before reading a command. If at any time a new-line is typed and further input is needed to complete a command, then the secondary prompt (that is, the value of `PS2`) is issued.

**Conditional Expressions**

A *conditional expression* is used with the `[ [ ] compound command to test attributes of files and to compare strings. Word splitting and file name generation are not performed on the words between `[` and `]`. Each expression can be constructed from one or more of the following unary or binary expressions:

- `a file` True, if file exists.
- `b file` True, if file exists and is a block special file.
- `c file` True, if file exists and is a character special file.
True, if file exists and is a directory.

True, if file exists.

True, if file exists and is an ordinary file.

True, if file exists and is has its setgid bit set.

True, if file exists and is has its sticky bit set.

True, if length of string is non-zero.

True, if option named option is on.

True, if file exists and is a fifo special file or a pipe.

True, if file exists and is readable by current process.

True, if file exists and has size greater than zero.

True, if file descriptor number fildes is open and associated with a terminal device.

True, if file exists and has its setuid bit set.

True, if file exists and is writable by current process.

True, if file exists and is executable by current process. If file exists and is a directory, then the current process has permission to search in the directory.

True, if length of string is zero.

True, if file exists and is a symbolic link.

True, if file exists and is owned by the effective user id of this process.

True, if file exists and its group matches the effective group id of this process.

True, if file exists and is a socket.

True, if file1 exists and is newer than file2.

True, if file1 exists and is older than file2.

True, if file1 and file2 exist and refer to the same file.

True if the string string is not the null string.

True, if string matches pattern.

True, if string does not match pattern.

True if the strings string1 and string2 are identical.

True if the strings string1 and string2 are not identical.
string1 < string2  True, if string1 comes before string2 based on strings interpreted as appropriate to the locale setting for category LC_COLLATE.

string1 > string2  True, if string1 comes after string2 based on strings interpreted as appropriate to the locale setting for category LC_COLLATE.

exp1 -eq exp2  True, if exp1 is equal to exp2.

exp1 -ne exp2  True, if exp1 is not equal to exp2.

exp1 -lt exp2  True, if exp1 is less than exp2.

exp1 -gt exp2  True, if exp1 is greater than exp2.

exp1 -le exp2  True, if exp1 is less than or equal to exp2.

exp1 -ge exp2  True, if exp1 is greater than or equal to exp2.

In each of the above expressions, if file is of the form /dev/fd/n, where n is an integer, then the test is applied to the open file whose descriptor number is n.

A compound expression can be constructed from these primitives by using any of the following, listed in decreasing order of precedence.

(expression)  True, if expression is true. Used to group expressions.

! expression  True if expression is false.

expression1 && expression2  True, if expression1 and expression2 are both true.

expression1 || expression2  True, if either expression1 or expression2 is true.

Before a command is executed, its input and output may be redirected using a special notation interpreted by the shell. The following may appear anywhere in a simple-command or may precede or follow a command and are not passed on to the invoked command. Command and parameter substitution occur before word or digit is used except as noted below. File name generation occurs only if the pattern matches a single file, and blank interpretation is not performed.

<word  Use file word as standard input (file descriptor 0).

>word  Use file word as standard output (file descriptor 1). If the file does not exist then it is created. If the file exists, and the noclobber option is on, this causes an error; otherwise, it is truncated to zero length.

>|word  Same as >, except that it overrides the noclobber option.
Use file \textit{word} as standard output. If the file exists, output is appended to it (by first seeking to the EOF). Otherwise, the file is created.

Open file \textit{word} for reading and writing as standard input.

The shell input is read up to a line that is the same as \textit{word}, or to an EOF. No parameter substitution, command substitution, or file name generation is performed on \textit{word}. The resulting document, called a \textit{here-document}, becomes the standard input. If any character of \textit{word} is quoted, no interpretation is placed upon the characters of the document. Otherwise, parameter and command substitution occur, \texttt{\textbackslash NEWLINE} is ignored, and \texttt{\textbackslash} must be used to quote the characters \texttt{\textbackslash} , \texttt{\$} , \texttt{'} , and the first character of \textit{word}. If \texttt{-} is appended to \texttt{<<}, then all leading tabs are stripped from \textit{word} and from the document.

The standard input is duplicated from file descriptor \texttt{digit} (see \texttt{dup(2)}). Similarly for the standard output using \texttt{>>&digit}.

The standard input is closed. Similarly for the standard output using \texttt{>&--}.

The input from the co-process is moved to standard input.

The output to the co-process is moved to standard output.

If one of the above is preceded by a digit, then the file descriptor number referred to is that specified by the digit (instead of the default 0 or 1). For example:

\begin{verbatim}
... 2>&1
\end{verbatim}

means file descriptor 2 is to be opened for writing as a duplicate of file descriptor 1.

The order in which redirections are specified is significant. The shell evaluates each redirection in terms of the (file descriptor, file) association at the time of evaluation. For example:

\begin{verbatim}
... 1>&fname 2>&1
\end{verbatim}

first associates file descriptor 1 with file \textit{fname}. It then associates file descriptor 2 with the file associated with file descriptor 1 (that is, \textit{fname}). If the order of redirections were reversed, file descriptor 2 would be associated with the terminal (assuming file descriptor 1 had been) and then file descriptor 1 would be associated with file \textit{fname}. 
If a command is followed by & and job control is not active, then the default standard input for the command is the empty file /dev/null. Otherwise, the environment for the execution of a command contains the file descriptors of the invoking shell as modified by input/output specifications.

The environment (see environ(5)) is a list of name-value pairs that is passed to an executed program in the same way as a normal argument list. The names must be identifiers and the values are character strings. The shell interacts with the environment in several ways. On invocation, the shell scans the environment and creates a variable for each name found, giving it the corresponding value and marking it export. Executed commands inherit the environment. If the user modifies the values of these variables or creates new ones, using the export or typeset -x commands, they become part of the environment. The environment seen by any executed command is thus composed of any name-value pairs originally inherited by the shell, whose values may be modified by the current shell, plus any additions which must be noted in export or typeset -x commands.

The environment for any simple-command or function may be augmented by prefixing it with one or more variable assignments. A variable assignment argument is a word of the form identifier=value. Thus:

```
TERM=450 cmd args
```

and

```
(export TERM; TERM=450; cmd args)
```

are equivalent (as far as the above execution of cmd is concerned, except for special commands listed below that are preceded with an asterisk).

If the -k flag is set, all variable assignment arguments are placed in the environment, even if they occur after the command name. The following first prints a=b c and then c:

```
echo a=b c
set -k echo
a=b c
```

This feature is intended for use with scripts written for early versions of the shell and its use in new scripts is strongly discouraged. It is likely to disappear someday.

The function reserved word, described in the Commands section above, is used to define shell functions. Shell functions are read in and stored internally. Alias names are resolved when the function is read. Functions are executed like commands with the arguments passed as positional parameters. (See Execution below.)
Functions execute in the same process as the caller and share all files and present working directory with the caller. Traps caught by the caller are reset to their default action inside the function. A trap condition that is not caught or ignored by the function causes the function to terminate and the condition to be passed on to the caller.

A trap on EXIT set inside a function is executed after the function completes in the environment of the caller. This is true only for non-POSIX-style functions, that is, functions declared as

```
function func
```

as opposed to POSIX-style functions, declared as

```
func()
```

Ordinarily, variables are shared between the calling program and the function. However, the `typeset` special command used within a function defines local variables whose scope includes the current function and all functions it calls.

The special command `return` is used to return from function calls. Errors within functions return control to the caller.

The names of all functions can be listed with `typeset+`. `typeset -` lists all function names as well as the text of all functions. `typeset - function-names` lists the text of the named functions only. Functions can be undefined with the `-f` option of the `unset` special command.

Ordinarily, functions are `unset` when the shell executes a shell script. The `-xf` option of the `typeset` command allows a function to be exported to scripts that are executed without a separate invocation of the shell. Functions that need to be defined across separate invocations of the shell should be specified in the `ENV` file with the `-xf` option of `typeset`.

A function is a user-defined name that is used as a simple command to call a compound command with new positional parameters. A function is defined with a `function definition command`.

The format of a function definition command is as follows:

```
fname () compound-command[io-redirect ...]
```

The function is named `fname`; it must be a name. An implementation may allow other characters in a function name as an extension. The implementation will maintain separate name spaces for functions and variables.

The `()` in the function definition command consists of two operators. Therefore, intermixing blank characters with the `fname`, `,` and `)` is allowed, but unnecessary.

The argument `compound-command` represents a compound command.
When the function is declared, none of the expansions in `wordexp` will be performed on the text in `compound-command` or `io-redirect`; all expansions will be performed as normal each time the function is called. Similarly, the optional `io-redirect` redirections and any variable assignments within `compound-command` will be performed during the execution of the function itself, not the function definition.

When a function is executed, it will have the syntax-error and variable-assignment properties described for the special built-in utilities.

The `compound-command` will be executed whenever the function name is specified as the name of a simple command. The operands to the command temporarily will become the positional parameters during the execution of the `compound-command`; the special parameter `#` will also be changed to reflect the number of operands. The special parameter `0` will be unchanged. When the function completes, the values of the positional parameters and the special parameter `#` will be restored to the values they had before the function was executed. If the special built-in `return` is executed in the `compound-command`, the function will complete and execution will resume with the next command after the function call.

An example of how a function definition can be used wherever a simple command is allowed:

```bash
# If variable i is equal to "yes", # define function foo to be ls -l
#
[ "$i" = yes ] && foo() {
   ls -l
}
```

The exit status of a function definition will be 0 if the function was declared successfully; otherwise, it will be greater than zero. The exit status of a function invocation will be the exit status of the last command executed by the function.

### Jobs

If the `monitor` option of the `set` command is turned on, an interactive shell associates a job with each pipeline. It keeps a table of current jobs, printed by the `jobs` command, and assigns them small integer numbers. When a job is started asynchronously with `&`, the shell prints a line which looks like:

```
[1] 1234
```

indicating that the job, which was started asynchronously, was job number 1 and had one (top-level) process, whose process id was 1234.

If you are running a job and wish to do something else you may press the key `^Z` (Control-Z) which sends a STOP signal to the current job. The shell will then normally indicate that the job has been ‘Stopped’, and print another prompt. You can then manipulate the state of this job, putting it in the background with the `bg` command, or run some other commands and then eventually bring the job back into the foreground with the foreground command `fg`. A `^Z` takes effect immediately and is like an interrupt in that pending output and unread input are discarded when it is typed.
A job being run in the background will stop if it tries to read from the terminal. Background jobs are normally allowed to produce output, but this can be disabled by giving the command "stty tostop". If you set this tty option, then background jobs will stop when they try to produce output as they do when they try to read input.

There are several ways to refer to jobs in the shell. A job can be referred to by the process id of any process of the job or by one of the following:

- `%number` The job with the given number.
- `%string` Any job whose command line begins with `string`.
- `?string` Any job whose command line contains `string`.
- `%%` Current job.
- `%+` Equivalent to `%%`.
- `%-` Previous job.

The shell learns immediately whenever a process changes state. It normally informs you whenever a job becomes blocked so that no further progress is possible, but only just before it prints a prompt. This is done so that it does not otherwise disturb your work.

When the monitor mode is on, each background job that completes triggers any trap set for CHLD.

When you try to leave the shell while jobs are running or stopped, you will be warned with the message, ‘You have stopped(running) jobs.’ You may use the `jobs` command to see what they are. If you do this or immediately try to exit again, the shell will not warn you a second time, and the stopped jobs will be terminated. If you have `nohup`'ed jobs running when you attempt to logout, you will be warned with the message:

You have jobs running.

You will then need to logout a second time to actually logout. However, your background jobs will continue to run.

**Signals**

The INT and QUIT signals for an invoked command are ignored if the command is followed by `&` and the `monitor` option is not active. Otherwise, signals have the values inherited by the shell from its parent (but see also the `trap` special command below).

**Execution**

Each time a command is executed, the above substitutions are carried out. If the command name matches one of the `Special Commands` listed below, it is executed within the current shell process. Next, the command name is checked to see if it matches one of the user defined functions. If it does, the positional parameters are saved and then reset to the arguments of the function call. When the function completes or issues a `return`, the positional parameter list is restored and any trap set on `EXIT` within the function is executed. The value of a function is the value of the
last command executed. A function is also executed in the current shell process. If a command name is not a special command or a user defined function, a process is created and an attempt is made to execute the command via exec(2).

The shell variable PATH defines the search path for the directory containing the command. Alternative directory names are separated by a colon (:). The default path is /bin:/usr/bin: (specifying /bin, /usr/bin, and the current directory in that order). The current directory can be specified by two or more adjacent colons, or by a colon at the beginning or end of the path list. If the command name contains a / then the search path is not used. Otherwise, each directory in the path is searched for an executable file. If the file has execute permission but is not a directory or an a.out file, it is assumed to be a file containing shell commands. A sub-shell is spawned to read it. All non-exported aliases, functions, and variables are removed in this case. A parenthesized command is executed in a sub-shell without removing non-exported quantities.

The text of the last HISTSIZE (default 128) commands entered from a terminal device is saved in a history file. The file $HOME/.sh_history is used if the HISTFILE variable is not set or if the file it names is not writable. A shell can access the commands of all interactive shells which use the same named HISTFILE. The special command fc is used to list or edit a portion of this file. The portion of the file to be edited or listed can be selected by number or by giving the first character or characters of the command. A single command or range of commands can be specified. If you do not specify an editor program as an argument to fc then the value of the variable FCEDIT is used. If FCEDIT is not defined, then /bin/ed is used. The edited command(s) is printed and re-executed upon leaving the editor. The editor name – is used to skip the editing phase and to re-execute the command. In this case a substitution parameter of the form old=new can be used to modify the command before execution. For example, if r is aliased to ‘fc -e ‘ then typing ‘r bad=good c’ will re-execute the most recent command which starts with the letter c, replacing the first occurrence of the string bad with the string good.

Normally, each command line entered from a terminal device is simply typed followed by a new-line (RETURN or LINEFEED). If either the emacs, gmacs, or vi option is active, the user can edit the command line. To be in either of these edit modes set the corresponding option. An editing option is automatically selected each time the VISUAL or EDITOR variable is assigned a value ending in either of these option names.

The editing features require that the user’s terminal accept RETURN as carriage return without line feed and that a space must overwrite the current character on the screen.

The editing modes implement a concept where the user is looking through a window at the current line. The window width is the value of COLUMNS if it is defined, otherwise 80. If the window width is too small to display the prompt and leave at least 8 columns to enter input, the prompt is truncated from the left. If the line is longer than the window width minus two, a mark is displayed at the end of the window to
notify the user. As the cursor moves and reaches the window boundaries the window
will be centered about the cursor. The mark is a > if the line extends on the right side
of the window, < if the line extends on the left, and * if the line extends on both sides
of the window.

The search commands in each edit mode provide access to the history file. Only
strings are matched, not patterns, although a leading caret (\^) in the string restricts the
match to begin at the first character in the line.

This mode is entered by enabling either the emacs or gmacs option. The only
difference between these two modes is the way they handle \^T. To edit, move the
cursor to the point needing correction and then insert or delete characters or words as
needed. All the editing commands are control characters or escape sequences. The
notation for control characters is caret (\^) followed by the character. For example, \^F
is the notation for control F. This is entered by depressing 'f' while holding down the
CTRL (control) key. The SHIFT key is not depressed. (The notation \^? indicates the
DEL (delete) key.)

The notation for escape sequences is \^\- followed by a character. For example, \^F
(pronounced Meta f) is entered by depressing ESC (ascii 033) followed by 'f'. (\^\-f
would be the notation for ESC followed by SHIFT (capital) f.)

All edit commands operate from any place on the line (not just at the beginning).
Neither the RETURN nor the LINEFEED key is entered after edit commands except
when noted.

\^F  Move cursor forward (right) one character.
\^B  Move cursor backward (left) one character.
\M-\f Move cursor forward one word. (The emacs editor's idea of a
      word is a string of characters consisting of only letters, digits and
      underscores.)
\A  Move cursor to start of line.
\E  Move cursor to end of line.
\} char Move cursor forward to character char on current line.
\M-\} char Move cursor backward to character char on current line.
\^X\^X Interchange the cursor and mark.
\erase (User defined erase character as defined by the \texttt{stty}(1) command,
      usually \^H or #.) Delete previous character.
\D  Delete current character.
\M-\d Delete current word.
\M-\^H (Meta-backspace) Delete previous word.
<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M-h</td>
<td>Delete previous word.</td>
</tr>
<tr>
<td>M-^?</td>
<td>(Meta-DEL) Delete previous word if your interrupt character is ^? (DEL, the default) then this command will not work.</td>
</tr>
<tr>
<td>^T</td>
<td>Transpose current character with next character in emacs mode.</td>
</tr>
<tr>
<td></td>
<td>Transpose two previous characters in gmacs mode.</td>
</tr>
<tr>
<td>^C</td>
<td>Capitalize current character.</td>
</tr>
<tr>
<td>M-c</td>
<td>Capitalize current word.</td>
</tr>
<tr>
<td>M-l</td>
<td>Change the current word to lower case.</td>
</tr>
<tr>
<td>^K</td>
<td>Delete from the cursor to the end of the line. If preceded by numerical parameter whose value is less than the current cursor position, then delete from given position up to the cursor. If preceded by a numerical parameter whose value is greater than the current cursor position, then delete from cursor up to given cursor position.</td>
</tr>
<tr>
<td>^W</td>
<td>Kill from the cursor to the mark.</td>
</tr>
<tr>
<td>M-p</td>
<td>Push the region from the cursor to the mark on the stack.</td>
</tr>
<tr>
<td>kill</td>
<td>(User defined kill character as defined by the stty(1) command, usually ^G or @.) Kill the entire current line. If two kill characters are entered in succession, all kill characters from then on cause a line feed (useful when using paper terminals).</td>
</tr>
<tr>
<td>^Y</td>
<td>Restore last item removed from line. (Yank item back to the line.)</td>
</tr>
<tr>
<td>^L</td>
<td>Line feed and print current line.</td>
</tr>
<tr>
<td>^@</td>
<td>(null character) Set mark.</td>
</tr>
<tr>
<td>M-space</td>
<td>(Meta space) Set mark.</td>
</tr>
<tr>
<td>J</td>
<td>(New line) Execute the current line.</td>
</tr>
<tr>
<td>M</td>
<td>(Return) Execute the current line.</td>
</tr>
<tr>
<td>eof</td>
<td>End-of-file character, normally ^D, is processed as an End-of-file only if the current line is null.</td>
</tr>
<tr>
<td>^P</td>
<td>Fetch previous command. Each time ^P is entered the previous command back in time is accessed. Moves back one line when not on the first line of a multi-line command.</td>
</tr>
<tr>
<td>M-&lt;</td>
<td>Fetch the least recent (oldest) history line.</td>
</tr>
<tr>
<td>M-&gt;</td>
<td>Fetch the most recent (youngest) history line.</td>
</tr>
<tr>
<td>^N</td>
<td>Fetch next command line. Each time ^N is entered the next command line forward in time is accessed.</td>
</tr>
</tbody>
</table>
Reverse search history for a previous command line containing *string*. If a parameter of zero is given, the search is forward. *string* is terminated by a RETURN or NEW LINE. If *string* is preceded by a ^, the matched line must begin with *string*. If *string* is omitted, then the next command line containing the most recent *string* is accessed. In this case a parameter of zero reverses the direction of the search.

Operate. Execute the current line and fetch the next line relative to current line from the history file.

(Escape) Define numeric parameter, the digits are taken as a parameter to the next command. The commands that accept a parameter are ^F, ^B, erase, ^C, ^D, ^K, ^R, ^P, ^N, ^], M-_, M-^], M-_b, M-_c, M-_d, M-_f, M-_h, M-_l and M-_^H.

Soft-key. Your alias list is searched for an alias by the name _letter and if an alias of this name is defined, its value will be inserted on the input queue. The letter must not be one of the above meta-functions.

Soft-key. Your alias list is searched for an alias by the name ___letter and if an alias of this name is defined, its value will be inserted on the input queue. The can be used to program functions keys on many terminals.

The last word of the previous command is inserted on the line. If preceded by a numeric parameter, the value of this parameter determines which word to insert rather than the last word.

Same as M-_.

An asterisk is appended to the end of the word and a file name expansion is attempted.

File name completion. Replaces the current word with the longest common prefix of all filenames matching the current word with an asterisk appended. If the match is unique, a / is appended if the file is a directory and a space is appended if the file is not a directory.

List files matching current word pattern if an asterisk were appended.

Multiply parameter of next command by 4.

Escape next character. Editing characters, the user’s erase, kill and interrupt (normally ^?) characters may be entered in a command line or in a search string if preceded by a \ . The \ removes the next character’s editing features (if any).

Display version of the shell.
vi Editing Mode

There are two typing modes. Initially, when you enter a command you are in the input mode. To edit, enter control mode by typing ESC (033) and move the cursor to the point needing correction and then insert or delete characters or words as needed. Most control commands accept an optional repeat count prior to the command.

When in vi mode on most systems, canonical processing is initially enabled and the command will be echoed again if the speed is 1200 baud or greater and it contains any control characters or less than one second has elapsed since the prompt was printed. The ESC character terminates canonical processing for the remainder of the command and the user can then modify the command line. This scheme has the advantages of canonical processing with the type-ahead echoing of raw mode.

If the option viraw is also set, the terminal will always have canonical processing disabled. This mode is implicit for systems that do not support two alternate end of line delimiters, and may be helpful for certain terminals.

Input Edit Commands

By default the editor is in input mode.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>erase</code></td>
<td>(User defined erase character as defined by the <code>stty</code> command, usually <code>^H</code> or <code>#</code>) Delete previous character.</td>
</tr>
<tr>
<td><code>^w</code></td>
<td>Delete the previous blank separated word.</td>
</tr>
<tr>
<td><code>^D</code></td>
<td>Terminate the shell.</td>
</tr>
<tr>
<td><code>^V</code></td>
<td>Escape next character. Editing characters and the user's erase or kill characters may be entered in a command line or in a search string if preceded by a <code>^V</code>. The <code>^V</code> removes the next character's editing features (if any).</td>
</tr>
<tr>
<td><code>\</code></td>
<td>Escape the next erase or kill character.</td>
</tr>
</tbody>
</table>

Motion Edit Commands

These commands will move the cursor.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[count]</code></td>
<td>Cursor forward (right) one character.</td>
</tr>
<tr>
<td><code>[count]w</code></td>
<td>Cursor forward one alpha-numeric word.</td>
</tr>
<tr>
<td><code>[count]a</code></td>
<td>Cursor to the beginning of the next word that follows a blank.</td>
</tr>
<tr>
<td><code>[count]e</code></td>
<td>Cursor to end of word.</td>
</tr>
<tr>
<td><code>[count]b</code></td>
<td>Cursor to end of the current blank delimited word.</td>
</tr>
<tr>
<td><code>[count]h</code></td>
<td>Cursor backward (left) one character.</td>
</tr>
<tr>
<td><code>[count]d</code></td>
<td>Cursor backward one word.</td>
</tr>
<tr>
<td><code>[count]B</code></td>
<td>Cursor to preceding blank separated word.</td>
</tr>
<tr>
<td><code>[count]l</code></td>
<td>Cursor to column count.</td>
</tr>
</tbody>
</table>
Find the next character \( c \) in the current line.

Find the previous character \( c \) in the current line.

Equivalent to \( f \) followed by \( h \).

Equivalent to \( F \) followed by \( l \).

Repeats \( count \) times, the last single character find command, \( f \), \( F \), \( t \), or \( T \).

Reverses the last single character find command \( count \) times.

Cursor to start of line.

Cursor to first non-blank character in line.

Cursor to end of line.

Moves to balancing \( ( \), \( ) \), \( \{ \), \( \} \), or \( [ \). If cursor is not on one of the above characters, the remainder of the line is searched for the first occurrence of one of the above characters first.

These commands access your command history.

Fetch previous command. Each time \( k \) is entered the previous command back in time is accessed.

Equivalent to \( k \).

Fetch next command. Each time \( j \) is entered, the next command forward in time is accessed.

Equivalent to \( j \).

The command number \( count \) is fetched. The default is the least recent history command.

Search backward through history for a previous command containing \( string \). \( string \) is terminated by a RETURN or NEWLINE. If \( string \) is preceded by a \( ^ \), the matched line must begin with \( string \). If \( string \) is NULL, the previous string will be used.

Same as \( / \) except that search will be in the forward direction.

Search for next match of the last pattern to \( / \) or \( ? \) commands.

Search for next match of the last pattern to \( / \) or \( ? \), but in reverse direction. Search history for the \( string \) entered by the previous \( / \) command.

These commands will modify the line.

Enter input mode and enter text after the current character.

Append text to the end of the line. Equivalent to \( $a \).
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[count]c[motion]</td>
<td>Delete current character through the character that \textit{motion} would move the cursor to and enter input mode. If \textit{motion} is \textit{c}, the entire line will be deleted and input mode entered.</td>
</tr>
<tr>
<td>\textit{C}</td>
<td>Delete the current character through the end of line and enter input mode. Equivalent to \textit{cs}.</td>
</tr>
<tr>
<td>[count]s</td>
<td>Delete \textit{count} characters and enter input mode. Equivalent to \textit{cc}.</td>
</tr>
<tr>
<td>\textit{D}</td>
<td>Delete the current character through the end of line. Equivalent to \textit{d$}.</td>
</tr>
<tr>
<td>[count]d[motion]</td>
<td>Delete current character through the character that \textit{motion} would move to. If \textit{motion} is \textit{d}, the entire line will be deleted.</td>
</tr>
<tr>
<td>\textit{i}</td>
<td>Enter input mode and insert text before the current character.</td>
</tr>
<tr>
<td>\textit{I}</td>
<td>Insert text before the beginning of the line. Equivalent to 0\textit{i}.</td>
</tr>
<tr>
<td>[count]P</td>
<td>Place the previous text modification before the cursor.</td>
</tr>
<tr>
<td>[count]p</td>
<td>Place the previous text modification after the cursor.</td>
</tr>
<tr>
<td>\textit{R}</td>
<td>Enter input mode and replace characters on the screen with characters you type overlay fashion.</td>
</tr>
<tr>
<td>[count]xc</td>
<td>Replace the \textit{count} character(s) starting at the current cursor position with \textit{c}, and advance the cursor.</td>
</tr>
<tr>
<td>[count]x</td>
<td>Delete current character.</td>
</tr>
<tr>
<td>[count]X</td>
<td>Delete preceding character.</td>
</tr>
<tr>
<td>[count].</td>
<td>Repeat the previous text modification command.</td>
</tr>
<tr>
<td>[count]-</td>
<td>Invert the case of the \textit{count} character(s) starting at the current cursor position and advance the cursor.</td>
</tr>
<tr>
<td>[count]_</td>
<td>Causes the \textit{count} word of the previous command to be appended and input mode entered. The last word is used if \textit{count} is omitted.</td>
</tr>
<tr>
<td>*</td>
<td>Causes an * to be appended to the current word and file name generation attempted. If no match is found, it rings the bell. Otherwise, the word is replaced by the matching pattern and input mode is entered.</td>
</tr>
</tbody>
</table>
Filename completion. Replaces the current word with the longest common prefix of all filenames matching the current word with an asterisk appended. If the match is unique, a / is appended if the file is a directory and a space is appended if the file is not a directory.

Miscellaneous commands.

[count]ymotion
γ[count]motion
Yank current character through character that motion would move the cursor to and puts them into the delete buffer. The text and cursor are unchanged.

Y
Yanks from current position to end of line. Equivalent to γ$.

u
Undo the last text modifying command.

U
Undo all the text modifying commands performed on the line.

[count]v
Returns the command fc -e
$[{VISUAL:=-${EDITOR:-vi}}] count in the input buffer. If count is omitted, then the current line is used.

^L
Line feed and print current line. Has effect only in control mode.

J
(New line) Execute the current line, regardless of mode.

M
(Return) Execute the current line, regardless of mode.

#
If the first character of the command is a #, then this command deletes this # and each # that follows a newline. Otherwise, sends the line after inserting a # in front of each line in the command. Useful for causing the current line to be inserted in the history as a comment and removing comments from previous comment commands in the history file.

=
List the file names that match the current word if an asterisk were appended it.

@letter
Your alias list is searched for an alias by the name _letter and if an alias of this name is defined, its value will be inserted on the input queue for processing.

The following simple-commands are executed in the shell process. Input/Output redirection is permitted. Unless otherwise indicated, the output is written on file descriptor 1 and the exit status, when there is no syntax error, is 0. Commands that are preceded by one or two * (asterisks) are treated specially in the following ways:
1. Variable assignment lists preceding the command remain in effect when the command completes.

2. I/O redirections are processed after variable assignments.

3. Errors cause a script that contains them to abort.

4. Words, following a command preceded by `*` that are in the format of a variable assignment, are expanded with the same rules as a variable assignment. This means that tilde substitution is performed after the `=` sign and word splitting and file name generation are not performed.

`: [ arg ... ]

The command only expands parameters.

`. file [ arg ... ]

Read the complete file then execute the commands. The commands are executed in the current shell environment. The search path specified by `PATH` is used to find the directory containing file. If any arguments arg are given, they become the positional parameters. Otherwise the positional parameters are unchanged. The exit status is the exit status of the last command executed.

** alias [-tx] [name[ =value ]] ...

alias with no arguments prints the list of aliases in the form name=value on standard output. An alias is defined for each name whose value is given. A trailing space in value causes the next word to be checked for alias substitution. The `-t` flag is used to set and list tracked aliases. The value of a tracked alias is the full pathname corresponding to the given name. The value becomes undefined when the value of `PATH` is reset but the aliases remained tracked. Without the `-t` flag, for each name in the argument list for which no value is given, the name and value of the alias is printed. The `-x` flag is used to set or print exported aliases. An exported alias is defined for scripts invoked by name. The exit status is non-zero if a name is given, but no value, and no alias has been defined for the name.

`bg [ %job ... ]`

This command is only on systems that support job control. Puts each specified job into the background. The current job is put in the background if job is not specified. See "Jobs" section above for a description of the format of job.

* break [ n ]

Exit from the enclosed for, while, until, or select loop, if any. If n is specified then break n levels.

* continue [ n ]

Resume the next iteration of the enclosed for, while, until, or select loop. If n is specified then resume at the n-th enclosed loop.

cd [ arg ]

This command can be in either of two forms. In the first form it changes the current directory to arg. If arg is – the directory is changed to the previous directory. The shell variable `HOME` is the default arg. The variable `PWD` is set to the current directory. The shell variable `CDPATH` defines the search path for the directory.
containing arg. Alternative directory names are separated by a colon (:\). The
default path is null (specifying the current directory). Notice that the current
directory is specified by a null path name, which can appear immediately after the
equal sign or between the colon delimiters anywhere else in the path list. If arg
begins with a / then the search path is not used. Otherwise, each directory in the
path is searched for arg.

The second form of cd substitutes the string new for the string old in the current
directory name, PWD and tries to change to this new directory. The cd command
may not be executed by ksh.

command [-p] [command_name] [argument ...]
command [-v -V] command_name

The command utility causes the shell to treat the arguments as a simple command,
suppressing the shell function lookup. The -p flag performs the command search
using a default value for PATH that is guaranteed to find all of the standard utilities.
The -v flag writes a string to standard output that indicates the pathname or
command that will be used by the shell, in the current shell execution environment,
to invoke command_name. The -V flag writes a string to standard output that
indicates how the name given in the command_name operand will be interpreted by
the shell, in the current shell execution environment.

echo [ arg ... ]
See echo(1) for usage and description.

* eval [ arg ... ]
The arguments are read as input to the shell and the resulting command(s)
executed.

* exec [ arg ... ]
If arg is given, the command specified by the arguments is executed in place of this
shell without creating a new process. Input/output arguments may appear and
affect the current process. If no arguments are given the effect of this command is to
modify file descriptors as prescribed by the input/output redirection list. In this
case, any file descriptor numbers greater than 2 that are opened with this
mechanism are closed when invoking another program.

* exit [ n ]
Causes the calling shell or shell script to exit with the exit status specified by n. The
value will be the least significant 8 bits of the specified status. If n is omitted then
the exit status is that of the last command executed. When exit occurs when
executing a trap, the last command refers to the command that executed before the
trap was invoked. An EOF will also cause the shell to exit except for a shell which
has the ignoreeof option (See set below) turned on.

** export [ name=value ] ...
The given names are marked for automatic export to the environment of
subsequently-executed commands.

fc [-e ename ] [-nlz] [first[ last]]
fc -e - [ old=new ] [ command ]
In the first form, a range of commands from \texttt{first} to \texttt{last} is selected from the last \texttt{HISTSIZE} commands that were typed at the terminal. The arguments \texttt{first} and \texttt{last} may be specified as a number or as a string. A string is used to locate the most recent command starting with the given string. A negative number is used as an offset to the current command number. If the \texttt{-1} flag is selected, the commands are listed on standard output. Otherwise, the editor program \texttt{ename} is invoked on a file containing these keyboard commands. If \texttt{ename} is not supplied, then the value of the variable \texttt{FCEDIT} (default /bin/ed) is used as the editor. When editing is complete, the edited command(s) is executed. If \texttt{last} is not specified then it will be set to \texttt{first}. If \texttt{first} is not specified the default is the previous command for editing and \texttt{-16} for listing. The flag \texttt{-r} reverses the order of the commands and the flag \texttt{-n} suppresses command numbers when listing. In the second form the command is re-executed after the substitution \texttt{old=\textasciitilde new} is performed. If there is not a \texttt{command} argument, the most recent command typed at this terminal is executed.

\texttt{fg [ \$\texttt{job} \ldots ]}

This command is only on systems that support job control. Each \texttt{job} specified is brought to the foreground. Otherwise, the current job is brought into the foreground. See "Jobs" section above for a description of the format of \texttt{job}.

\texttt{getopts optstring name [ arg \ldots ]}

Checks \texttt{arg} for legal options. If \texttt{arg} is omitted, the positional parameters are used. An option argument begins with a \texttt{+} or a \texttt{-}. An option not beginning with \texttt{+} or \texttt{-} or the argument \texttt{-} ends the options. \texttt{optstring} contains the letters that \texttt{getopts} recognizes. If a letter is followed by a \texttt{\textasciitilde}, that option is expected to have an argument. The options can be separated from the argument by blanks.

\texttt{getopts} places the next option letter it finds inside variable \texttt{name} each time it is invoked with a \texttt{+} prepended when \texttt{arg} begins with a \texttt{+}. The index of the next \texttt{arg} is stored in \texttt{OPTIND}. The option argument, if any, gets stored in \texttt{OPTARG}.

A leading \texttt{\textasciitilde} in \texttt{optstring} causes \texttt{getopts} to store the letter of an invalid option in \texttt{OPTARG}, and to set \texttt{name} to \texttt{?} for an unknown option and to \texttt{\textasciitilde} when a required option is missing. Otherwise, \texttt{getopts} prints an error message. The exit status is non-zero when there are no more options. See \texttt{getoptcvt(1)} for usage and description.

\texttt{hash [ name \ldots ]}

For each \texttt{name}, the location in the search path of the command specified by \texttt{name} is determined and remembered by the shell. The \texttt{-r} option causes the shell to forget all remembered locations. If no arguments are given, information about remembered commands is presented. \texttt{Hits} is the number of times a command has been invoked by the shell process. \texttt{Cost} is a measure of the work required to locate a command in the search path. If a command is found in a "relative" directory in the search path, after changing to that directory, the stored location of that command is recalculated. Commands for which this will be done are indicated by an asterisk (*) adjacent to the \texttt{hits} information. \texttt{Cost} will be incremented when the recalculation is done.
jobs [-lnp ] [ %job ...]
   Lists information about each given job; or all active jobs if job is omitted. The -l flag lists process ids in addition to the normal information. The -n flag displays only jobs that have stopped or exited since last notified. The -p flag causes only the process group to be listed. See "Jobs" section above and jobs(1) for a description of the format of job.

kill [ -sig ] %job ...
kill [ -sig ] pid ...
kill -l
   Sends either the TERM (terminate) signal or the specified signal to the specified jobs or processes. Signals are either given by number or by names (as given in signal(3HEAD) stripped of the prefix "SIG" with the exception that SIGCHD is named CHLD). If the signal being sent is TERM (terminate) or HUP (hangup), then the job or process will be sent a CONT (continue) signal if it is stopped. The argument job can be the process id of a process that is not a member of one of the active jobs. See Jobs for a description of the format of job. In the second form, kill -l, the signal numbers and names are listed.

let arg ... 
   Each arg is a separate arithmetic expression to be evaluated. See the Arithmetic Evaluation section above, for a description of arithmetic expression evaluation.

   The exit status is 0 if the value of the last expression is non-zero, and 1 otherwise.

login argument ...
   Equivalent to 'exec login argument...'. See login(1) for usage and description.

* newgrp [ arg ...]
   Equivalent to exec /bin/newgrp arg ...

print [-Rnprsu] [arg ...]
   The shell output mechanism. With no flags or with flag - or --, the arguments are printed on standard output as described by echo(1). The exit status is 0, unless the output file is not open for writing.

   -n      Suppress NEWLINE from being added to the output.
   -R | -r   Raw mode. Ignore the escape conventions of echo. The -R option will print all subsequent arguments and options other than -n.
   -p      Write the arguments to the pipe of the process spawned with | & instead of standard output.
   -s      Write the arguments to the history file instead of standard output.
   -u [ n ] Specify a one digit file descriptor unit number n on which the output will be placed. The default is 1.

pwd
   Equivalent to print -r - $PWD.
**read** [ -prsu[ n ] ] [ name?prompt ] [ name ... ]

The shell input mechanism. One line is read and is broken up into fields using the characters in IFS as separators. The escape character, (\), is used to remove any special meaning for the next character and for line continuation. In raw mode, -r, the \ character is not treated specially. The first field is assigned to the first name, the second field to the second name, etc., with leftover fields assigned to the last name. The -p option causes the input line to be taken from the input pipe of a process spawned by the shell using |&. If the -s flag is present, the input will be saved as a command in the history file. The flag -u can be used to specify a one digit file descriptor unit n to read from. The file descriptor can be opened with the exec special command. The default value of n is 0. If name is omitted then REPLY is used as the default name. The exit status is 0 unless the input file is not open for reading or an EOF is encountered. An EOF with the -p option causes cleanup for this process so that another can be spawned. If the first argument contains a ?, the remainder of this word is used as a prompt on standard error when the shell is interactive. The exit status is 0 unless an EOF is encountered.

**readonly** [ name[=value] ] ...

The given names are marked readonly and these names cannot be changed by subsequent assignment.

*return* [ n ]

Causes a shell function or ‘.’ script to return to the invoking script with the return status specified by n. The value will be the least significant 8 bits of the specified status. If n is omitted then the return status is that of the last command executed. If return is invoked while not in a function or a ‘.’ script, then it is the same as an exit.

**set** [ +abCefhkmnopstuvx ] [ +o option ] ... [ +A name ] [ arg ... ]

The flags for this command have meaning as follows:

- **-A**  Array assignment. Unset the variable name and assign values sequentially from the list arg. If +A is used, the variable name is not unset first.

- **-a**  All subsequent variables that are defined are automatically exported.

- **-b**  Causes the shell to notify the user asynchronously of background job completions. The following message will be written to standard error:

  " [%d]%<space>%s%a
",  <job-number>, <current>, <status>, <job-name>

where the fields are as follows:

- **<current>**  The character + identifies the job that would be used as a default for the fg or bg utilities; this job can also be specified using the job_id %+ or %. The character - identifies the job that would become the default if the current default job were to exit; this job can also be specified using the job_id %-. For other jobs, this

User Commands  619
field is a space character. At most one job can be identified with + and at most one job can be identified with -. If there is any suspended job, then the current job will be a suspended job. If there are at least two suspended jobs, then the previous job will also be a suspended job.

<job-number> A number that can be used to identify the process group to the wait, fg, bg, and kill utilities. Using these utilities, the job can be identified by prefixing the job number with %.

<status> Unspecified.

<job-name> Unspecified.

When the shell notifies the user a job has been completed, it may remove the job’s process ID from the list of those known in the current shell execution environment. Asynchronous notification will not be enabled by default.

-C Prevent existing files from being overwritten by the shell’s > redirection operator. The >| redirection operator will override this noclobber option for an individual file.

e If a command has a non-zero exit status, execute the ERR trap, if set, and exit. This mode is disabled while reading profiles.

-f Disables file name generation.

-h Each command becomes a tracked alias when first encountered.

-k All variable assignment arguments are placed in the environment for a command, not just those that precede the command name.

-m Background jobs will run in a separate process group and a line will print upon completion. The exit status of background jobs is reported in a completion message. On systems with job control, this flag is turned on automatically for interactive shells.

-n Read commands and check them for syntax errors, but do not execute them. Ignored for interactive shells.

-o The following argument can be one of the following option names:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alleport</td>
<td>Same as -a.</td>
</tr>
<tr>
<td>errexit</td>
<td>Same as -e.</td>
</tr>
<tr>
<td>bgnice</td>
<td>All background jobs are run at a lower priority. This is the default mode.</td>
</tr>
<tr>
<td>emacs</td>
<td>Puts you in an emacs style in-line editor for command entry.</td>
</tr>
</tbody>
</table>
gmacs

Puts you in a gmacs style in-line editor for
cmd entry.

ignoreeof

The shell will not exit on EOF. The command exit
must be used.

keyword

Same as -k.

markdirs

All directory names resulting from file name
generation have a trailing / appended.

monitor

Same as -m.

clobber

Prevents redirection > from truncating existing files.
Require >> to truncate a file when turned on.
Equivalent to -C.

noexec

Same as -n.

noglob

Same as -f.

nolog

Do not save function definitions in history file.

notify

Equivalent to -b.

nounset

Same as -u.

privileged

Same as -p.

verbose

Same as -v.

trackall

Same as -h.

vi

Puts you in insert mode of a vi style in-line editor
until you hit escape character 033. This puts you in
control mode. A return sends the line.

viraw

Each character is processed as it is typed in vi
mode.

xtrace

Same as -x.

If no option name is supplied, the current option settings are printed.

-p

Disables processing of the $HOME/.profile file and uses the file
/etc/suid_profile instead of the ENV file. This mode is on
whenever the effective uid is not equal to the real uid, or when the
effective gid is not equal to the real gid. Turning this off causes the
effective uid and gid to be set to the real uid and gid.

-s

Sort the positional parameters lexicographically.

-t

Exit after reading and executing one command.

-u

Treat unset parameters as an error when substituting.

-v

Print shell input lines as they are read.
-x  Print commands and their arguments as they are executed.
—  Turns off -x and -v flags and stops examining arguments for flags.
Do not change any of the flags; useful in setting $1 to a value beginning
with -. If no arguments follow this flag then the positional parameters
are unset.
Using + rather than - causes these flags to be turned off. These flags can
also be used upon invocation of the shell. The current set of flags may be
found in $-. Unless -A is specified, the remaining arguments are
positional parameters and are assigned, in order, to $1 $2 ... If no
arguments are given, the names and values of all variables are printed
on the standard output.

*shift [n]
The positional parameters from $n+1 $n+1 ... are renamed $1 ... , default
n is 1. The parameter n can be any arithmetic expression that evaluates to a
non-negative number less than or equal to $#.

stop %jobid ...
stop pid ...
stop stops the execution of a background job(s) by using its jobid, or of any process
by using its pid. (see ps(1)).

suspend
Stops the execution of the current shell (but not if it is the login shell).

test expression
Evaluate conditional expressions. See Conditional Expressions section above
and test(1) for usage and description.

*times
Print the accumulated user and system times for the shell and for processes run
from the shell.

*trap [arg sig ...]
arg is a command to be read and executed when the shell receives signal(s) sig. arg
is scanned once when the trap is set and once when the trap is taken. sig can be
specified as a signal number or signal name. trap commands are executed in order
of signal number. Any attempt to set a trap on a signal number that was ignored on
entry to the current shell is ineffective.

If arg is -, the shell will reset each sig to the default value. If arg is null (""), the
shell will ignore each specified sig if it arises. Otherwise, arg will be read and
executed by the shell when one of the corresponding sigs arises. The action of the
trap will override a previous action (either default action or one explicitly set). The
value of $? after the trap action completes will be the value it had before the trap
was invoked.
**type** `name`...

For each `name`, indicate how it would be interpreted if used as a command name.

**typeset [±HLRZfilrtux][name=value]...**

Sets attributes and values for shell variables and functions. When `typeset` is invoked inside a function, a new instance of the variables `name` is created. The variables `value` and `type` are restored when the function completes. The following list of attributes may be specified:
This flag provides UNIX to host-name file mapping on non-UNIX machines.

Left justify and remove leading blanks from value. If n is non-zero it defines the width of the field; otherwise, it is determined by the width of the value of first assignment. When the variable is assigned to, it is filled on the right with blanks or truncated, if necessary, to fit into the field. Leading zeros are removed if the -Z flag is also set. The -R flag is turned off.

Right justify and fill with leading blanks. If n is non-zero it defines the width of the field, otherwise it is determined by the width of the value of first assignment. The field is left filled with blanks or truncated from the end if the variable is reassigned. The -L flag is turned off.

Right justify and fill with leading zeros if the first non-blank character is a digit and the -L flag has not been set. If n is non-zero it defines the width of the field; otherwise, it is determined by the width of the value of first assignment.

The names refer to function names rather than variable names. No assignments can be made and the only other valid flags are -t, -u, and -x. The flag -t turns on execution tracing for this function. The flag -u causes this function to be marked undefined. The FPATH variable will be searched to find the function definition when the function is referenced. The flag -x allows the function definition to remain in effect across shell procedures invoked by name.

Parameter is an integer. This makes arithmetic faster. If n is non-zero it defines the output arithmetic base; otherwise, the first assignment determines the output base.

All upper-case characters are converted to lower-case. The upper-case flag, -u is turned off.

The given names are marked readonly and these names cannot be changed by subsequent assignment.

Tags the variables. Tags are user definable and have no special meaning to the shell.

All lower-case characters are converted to upper-case characters. The lower-case flag, -l is turned off.

The given names are marked for automatic export to the environment of subsequently-executed commands.

The -i attribute can not be specified along with -R, -L, -Z, or -f.
Using + rather than – causes these flags to be turned off. If no name arguments are given but flags are specified, a list of names (and optionally the values) of the variables which have these flags set is printed. (Using + rather than – keeps the values from being printed.) If no names and flags are given, the names and attributes of all variables are printed.

```plaintext
ulimit [ -HSacdfnstv ] [ limit ]
```
Set or display a resource limit. The available resources limits are listed below. Many systems do not contain one or more of these limits. The limit for a specified resource is set when limit is specified. The value of limit can be a number in the unit specified below with each resource, or the value unlimited. The H and S flags specify whether the hard limit or the soft limit for the given resource is set. A hard limit cannot be increased once it is set. A soft limit can be increased up to the value of the hard limit. If neither the H or S options is specified, the limit applies to both. The current resource limit is printed when limit is omitted. In this case the soft limit is printed unless H is specified. When more that one resource is specified, then the limit name and unit is printed before the value.

- a Lists all of the current resource limits.
- c The number of 512-byte blocks on the size of core dumps.
- d The number of K-bytes on the size of the data area.
- f The number of 512-byte blocks on files written by child processes (files of any size may be read).
- n The number of file descriptors plus 1.
- s The number of K-bytes on the size of the stack area.
- t The number of seconds to be used by each process.
- v The number of K-bytes for virtual memory.

If no option is given, -f is assumed.

```plaintext
umask [ -S ] [ mask ]
```
The user file-creation mask is set to mask (see umask(2)). mask can either be an octal number or a symbolic value as described in chmod(1). If a symbolic value is given, the new umask value is the complement of the result of applying mask to the complement of the previous umask value. If mask is omitted, the current value of the mask is printed. The -S flag produces symbolic output.

```plaintext
unalias name . .
```
The aliases given by the list of names are removed from the alias list.

```plaintext
unset [ -f ] name . .
```
The variables given by the list of names are unassigned, that is, their values and attributes are erased. readonly variables cannot be unset. If the -f flag is set, then the names refer to function names. Unsetting ERRNO, LINENO, MAILCHECK, OPTARG, OPTIND, RANDOM, SECONDS, TMOUT, and _ removes their special meaning even if they are subsequently assigned to.
*wait [job]*

Wait for the specified *job* and report its termination status. If *job* is not given then all currently active child processes are waited for. The exit status from this command is that of the process waited for. See *Jobs* for a description of the format of *job*.

*whence [ -pv ] name ...*

For each *name*, indicate how it would be interpreted if used as a command name.

The `-v` flag produces a more verbose report.

The `-p` flag does a path search for *name* even if name is an alias, a function, or a reserved word.

**Invocation**

If the shell is invoked by `exec(2)`, and the first character of argument zero (`$0`) is `-`, then the shell is assumed to be a login shell and commands are read from /etc/profile and then from either .profile in the current directory or `$HOME/.profile`, if either file exists. Next, commands are read from the file named by performing parameter substitution on the value of the environment variable ENV if the file exists. If the `-s` flag is not present and *arg* is, then a path search is performed on the first *arg* to determine the name of the script to execute. The script *arg* must have read permission and any `setuid` and `setgid` settings will be ignored. If the script is not found on the path, *arg* is processed as if it named a builtin command or function. Commands are then read as described below; the following flags are interpreted by the shell when it is invoked:

- `-c` Read commands from the `command_string` operand. Set the value of special parameter 0 from the value of the `command_name` operand and the positional parameters ($1, $2, and so on) in sequence from the remaining *arg* operands. No commands will be read from the standard input.

- `-s` If the `-s` flag is present or if no arguments remain, commands are read from the standard input. Shell output, except for the output of the `Special Commands` listed above, is written to file descriptor 2.

- `-i` If the `-i` flag is present or if the shell input and output are attached to a terminal (as told by `ioctl(2)`), then this shell is *interactive*. In this case, TERM is ignored (so that `kill 0` does not kill an interactive shell) and INTR is caught and ignored (so that `wait` is interruptible). In all cases, QUIT is ignored by the shell.

- `-r` If the `-r` flag is present the shell is a restricted shell.

The remaining flags and arguments are described under the `set` command above.

**rksh Only**

`rksh` is used to set up login names and execution environments whose capabilities are more controlled than those of the standard shell. The actions of `rksh` are identical to those of `ksh`, except that the following are disallowed:

- changing directory (see `cd(1)`)
- setting the value of `SHELL`, `ENV`, or `PATH`
- specifying path or command names containing `/
redirecting output (>, >|, <>, and >>)  
changing group (see newgrp(1)).

The restrictions above are enforced after .profile and the ENV files are interpreted.

When a command to be executed is found to be a shell procedure, rksh invokes ksh to execute it. Thus, it is possible to provide to the end-user shell procedures that have access to the full power of the standard shell, while imposing a limited menu of commands; this scheme assumes that the end-user does not have write and execute permissions in the same directory.

The net effect of these rules is that the writer of the .profile has complete control over user actions, by performing guaranteed setup actions and leaving the user in an appropriate directory (probably not the login directory).

The system administrator often sets up a directory of commands (that is, /usr/rbin) that can be safely invoked by rksh.

Errors detected by the shell, such as syntax errors, cause the shell to return a non-zero exit status. Otherwise, the shell returns the exit status of the last command executed (see also the exit command above). If the shell is being used non-interactively then execution of the shell file is abandoned. Run time errors detected by the shell are reported by printing the command or function name and the error condition. If the line number that the error occurred on is greater than one, then the line number is also printed in square brackets ([]) after the command or function name.

For a non-interactive shell, an error condition encountered by a special built-in or other type of utility will cause the shell to write a diagnostic message to standard error and exit as shown in the following table:

<table>
<thead>
<tr>
<th>Error</th>
<th>Special Built-in</th>
<th>Other Utilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shell language syntax error</td>
<td>will exit</td>
<td>will exit</td>
</tr>
<tr>
<td>Utility syntax error (option or operand error)</td>
<td>will exit</td>
<td>will not exit</td>
</tr>
<tr>
<td>Redirection error</td>
<td>will exit</td>
<td>will not exit</td>
</tr>
<tr>
<td>Variable assignment error</td>
<td>will exit</td>
<td>will not exit</td>
</tr>
<tr>
<td>Expansion error</td>
<td>will exit</td>
<td>will exit</td>
</tr>
<tr>
<td>Command not found</td>
<td>n/a</td>
<td>may exit</td>
</tr>
<tr>
<td>Dot script not found</td>
<td>will exit</td>
<td>n/a</td>
</tr>
</tbody>
</table>

An expansion error is one that occurs when the shell expansions are carried out (for example, ${x:y}, because ! is not a valid operator); an implementation may treat these as syntax errors if it is able to detect them during tokenization, rather than during expansion.
If any of the errors shown as “will (may) exit” occur in a subshell, the subshell will (may) exit with a non-zero status, but the script containing the subshell will not exit because of the error.

In all of the cases shown in the table, an interactive shell will write a diagnostic message to standard error without exiting.

**USAGE**

See largefile(5) for the description of the behavior of ksh and rksh when encountering files greater than or equal to 2 Gbyte (\(2^{31}\) bytes).

**EXIT STATUS**

Each command has an exit status that can influence the behavior of other shell commands. The exit status of commands that are not utilities is documented in this section. The exit status of the standard utilities is documented in their respective sections.

If a command is not found, the exit status will be 127. If the command name is found, but it is not an executable utility, the exit status will be 126. Applications that invoke utilities without using the shell should use these exit status values to report similar errors.

If a command fails during word expansion or redirection, its exit status will be greater than zero.

When reporting the exit status with the special parameter `?`, the shell will report the full eight bits of exit status available. The exit status of a command that terminated because it received a signal will be reported as greater than 128.

**FILES**

/etc/profile
/etc/suid_profile
$HOME/.profile
/tmp/sh*
/dev/null

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>/usr/bin/rksh</th>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
<td></td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>/usr/xpg4/bin/ksh</th>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWxcu4</td>
<td></td>
</tr>
<tr>
<td>ATTRIBUTE TYPE</td>
<td>ATTRIBUTE VALUE</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------</td>
<td></td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
<td></td>
</tr>
</tbody>
</table>

**SEE ALSO**

cat(1), cd(1), chmod(1), cut(1), echo(1), env(1), getoptcvt(1), jobs(1), login(1), newgrp(1), paste(1), ps(1), shell_builtins(1), stty(1), test(1), vi(1), dup(2), exec(2), fork(2), ioctl(2), lseek(2), pipe(2), ulimit(2), umask(2), wait(2), rand(3C), signal(3C), a.out(4), profile(4), attributes(5), environ(5), largefile(5), signal(3HEAD), XPG4(5)


**WARNINGS**

The use of setuid shell scripts is strongly discouraged.

**NOTES**

If a command which is a tracked alias is executed, and then a command with the same name is installed in a directory in the search path before the directory where the original command was found, the shell will continue to exec the original command. Use the -t option of the alias command to correct this situation.

Some very old shell scripts contain a `^` as a synonym for the pipe character `|`.

Using the `fc` built-in command within a compound command will cause the whole command to disappear from the history file.

The built-in command `.file` reads the whole file before any commands are executed. Therefore, `alias` and `unalias` commands in the file will not apply to any functions defined in the file.

When the shell executes a shell script that attempts to execute a non-existent command interpreter, the shell returns an erroneous diagnostic message that the shell script file does not exist.
NAME
ktutil – Kerberos keytab maintenance utility

SYNOPSIS
/usr/bin/ktutil

DESCRIPTION
The ktutil command is an interactive command-line interface utility for managing the keylist in keytab files. You must read in a keytab’s keylist before you can manage it. Also, the user running the ktutil command must have read/write permissions on the keytab. For example, if a keytab is owned by root, which it typically is, ktutil must be run as root to have the appropriate permissions.

COMMANDS

- clear_list, clear
  Clears the current keylist.
- read_kt file, rkt file
  Reads a keytab into the current keylist. You must specify a keytab file to read.
- write_kt file, wkt file
  Writes the current keylist to a keytab file. You must specify a keytab file to write. If the keytab file already exists, the current keylist is appended to the existing keytab file.
- delete_entry number, delent number
  Deletes an entry from the current keylist. Specify the entry by the keylist slot number.
- list, l
  Lists the current keylist.
- list_request, lr
  Lists available requests (commands).
- quit, exit, q
  Exits utility.

EXAMPLES
EXAMPLE 1 Deleting a principal from a file

The following example deletes the host/denver@ACME.com principal from the /etc/krb5/krb5.keytab file. Notice that if you want to delete an entry from an existing keytab, you must first write the keylist to a temporary keytab and then overwrite the existing keytab with the temporary keytab. This is because the wkt command actually appends the current keylist to an existing keytab, so you can’t use it to overwrite a keytab.

```bash
example# /usr/krb5/bin/ktutil
ktutil: rkt /etc/krb5/krb5.keytab
ktutil: list
--- --- ---------------------------------------
1 8 host/vail@ACME.COM
2 5 host/denver@ACME.COM
ktutil:delent 2
ktutil:l
--- --- ---------------------------------------
1 8 host/vail@ACME.COM
ktutil:wkt /tmp/krb5.keytab
ktutil:q
example# mv /tmp/krb5.keytab /etc/krb5/krb5.keytab
```

630 man pages section 1: User Commands • Last Revised 6 Nov 2000
EXAMPLE 1 Deleting a principal from a file

(Continued)

FILES
/etc/krb5/krb5.keytab keytab file for Kerberos clients

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWkrbu</td>
</tr>
</tbody>
</table>

SEE ALSO
SEAM(5)
The `last` command looks in the `/var/adm/wtmpx` file, which records all logins and logouts, for information about a user, a terminal, or any group of users and terminals. Arguments specify names of users or terminals of interest. If multiple arguments are given, the information applicable to any of the arguments is printed. For example, `last root console` lists all of root’s sessions, as well as all sessions on the console terminal. `last` displays the sessions of the specified users and terminals, most recent first, indicating the times at which the session began, the duration of the session, and the terminal on which the session took place. `last` also indicates whether the session is continuing or was cut short by a reboot.

The pseudo-user `reboot` logs in when the system reboots. Thus, `last reboot` will give an indication of mean time between reboots.

`last` with no arguments displays a record of all logins and logouts, in reverse order.

If `last` is interrupted, it indicates how far the search has progressed in `/var/adm/wtmpx`. If interrupted with a quit signal (generated by a CTRL-`), `last` indicates how far the search has progressed, and then continues the search.

The following options are supported:

- `-a` Displays the hostname in the last column.
- `-f filename` Uses `filename` as the name of the accounting file instead of `/var/adm/wtmpx`.
- `-n number | -number` Limits the number of entries displayed to that specified by `number`. These options are identical; the `-number` option is provided as a transition tool only and will be removed in future releases.

Date and time format is based on locale specified by the `LC_ALL`, `LC_TIME`, or `LANG` environments, in that order of priority.

The following attributes are defined:

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
</tbody>
</table>

The `last` command is used to display login and logout information about users and terminals.
SEE ALSO  utmpx(4), attributes(5)
lastcomm(1)

NAME    lastcomm – display the last commands executed, in reverse order

SYNOPSIS lastcomm [-f file] [-x] [command-name] ... [user-name] ... [terminal-name] ...

DESCRIPTION The lastcomm command gives information on previously executed commands. lastcomm with no arguments displays information about all the commands recorded during the current accounting file's lifetime. If called with arguments, lastcomm only displays accounting entries with a matching command-name, user-name, or terminal-name. If extended process accounting is active (see acctadm(1M)) and is recording the appropriate data items, lastcomm attempts to take data from the current extended process accounting file. If standard process accounting is active, lastcomm takes data from the current standard accounting file (see acct(2)).

If terminal-name is ‘- -’, there was no controlling TTY for the process. The process was probably executed during boot time. If terminal-name is ‘??’, the controlling TTY could not be decoded into a printable name.

For each process entry, lastcomm displays the following items of information:
- The command name under which the process was called.
- One or more flags indicating special information about the process. The flags have the following meanings:
  - F  The process performed a fork but not an exec.
  - S  The process ran as a set-user-id program.
- The name of the user who ran the process.
- The terminal which the user was logged in on at the time (if applicable).
- The amount of CPU time used by the process (in seconds).
- The date and time the process exited.

OPTIONS The following options are supported:
- -f file  Uses file as the source of accounting data. file may be either an extended process accounting file or a standard process accounting file.
- -x  Uses the currently active extended process accounting file. If extended processing accounting is inactive, no output will be produced.

EXAMPLES EXAMPLE 1 Listing executions of named commands
The command

```
example$ lastcomm a.out root term/01
```

produces a listing of all the executions of commands named a.out by user root while using the terminal term/01.
**EXAMPLE 2** Listing all user commands

The command

```bash
example$ lastcomm root
```

produces a listing of all the commands executed by user root.

**FILES**

- `/var/adm/pacct` standard accounting file
- `/var/adm/exacct/proc` extended accounting file

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
</tbody>
</table>

**SEE ALSO**

`last(1), acctadm(1M), acct(2), acct(3HEAD), sigvec(3UCB), core(4), attributes(5)`
The `ld` command combines relocatable object files, performs relocation, and resolves external symbols. `ld` operates in two modes, static or dynamic, as governed by the `-d` option. In all cases, the output of `ld` is left in a.out by default (see NOTES).

In static mode, `-dN`, relocatable object files given as arguments are combined to produce an executable object file. If the `-r` option is specified, relocatable object files are combined to produce one relocatable object file.

In dynamic mode, `-dy`, the default, relocatable object files given as arguments are combined to produce an executable object file that will be linked at execution with any shared object files given as arguments. If the `-G` option is specified, relocatable object files are combined to produce a shared object.

If any argument is a library, `ld` by default searches it exactly once at the point it encounters the library in the argument list. The library may be either a relocatable archive (see `ar(3HEAD)) or a shared object.

For an archive library, `ld` loads only those routines that define an unresolved external reference. `ld` searches the archive library symbol table sequentially with as many passes as are necessary to resolve external references that can be satisfied by library members. Thus, the order of members in the library is functionally unimportant, unless multiple library members exist that define the same external symbol. Archive libraries that have interdependencies may require multiple command-line definitions, or use of the `-z rescan` option.

A shared object consists of an indivisible, whole unit that has been generated by a previous link-edit of one or more input files. When the link-editor processes a shared object, the entire contents of the shared object become a logical part of the resulting
output file image. The shared object is not physically copied during the link-edit as its actual inclusion is deferred until process execution. This logical inclusion means that all symbol entries defined in the shared object are made available to the link-editing process.

No command-line option is required to distinguish 32-bit or 64-bit objects. The link-editor uses the ELF class of the first input relocatable object file it sees on the command line to govern the mode in which it will operate. Intermixing 32-bit and 64-bit objects is not permitted. See also the -64 option and the LD_NOEXEC_64 environment variable.

**OPTIONS**

The following options are supported:

- **-64**
  Creates a 64-bit object. By default, the class of the object being generated is determined from the first ELF object processed from the command line. This option is useful when creating an object directly with `ld` whose input is solely from a mapfile (see the `-M` option) or an archive library.

- **-a**
  In static mode only, produces an executable object file; gives errors for undefined references. This is the default behavior for static mode. `-a` may not be used with the `-r` option.

- **-b**
  In dynamic mode only, does no special processing for relocations that reference symbols in shared objects. Without the `-b` option, the link-editor creates special position-independent relocations for references to functions defined in shared objects and arranges for data objects defined in shared objects to be copied into the memory image of an executable by the runtime linker.

  The `-b` option is intended for specialized dynamic objects and is not recommended for general use. Its use suppresses all specialized processing required to insure an object’s shareability, and may even prevent the relocation of 64-bit executables.

- **-Bdirect**
  Establishes direct binding information by recording the relationship between each symbol reference and the dependency that provides the definition. The runtime linker uses this information to search directly for the symbol in the associated object rather than to carry out its default symbol search. Direct binding information can only be established to dependencies specified with the link-edit. Thus, you should use the `-z def` option. Objects that wish to interpose on symbols in a direct binding environment should identify themselves as interposers with the `-z interpose` option. The use of `-Bdirect` enables `-z lazyload` for all dependencies.

- **-Bdynamic | static**
  Options governing library inclusion. `-Bdynamic` is valid in dynamic mode only. These options may be specified any number of times on the command line as toggles: if the `-Bstatic` option is given, no shared objects will be accepted until `-Bdynamic` is seen. See also the `-l` option.
-B eliminate
Causes any global symbols not assigned to a version definition to be eliminated
from the symbol table. This option achieves the same symbol elimination as the
auto-elimination directive available as part of a mapfile version definition.

-B group
Establishes a shared object and its dependencies as a group. Objects within the
group will be bound to other members of the group at runtime. The runtime
processing of an object containing this flag mimics that which occurs if the object is
added to a process using dlopen(3DL) with the RTLD_GROUP mode. An object that
has an explicit dependency on an object identified as a group, will itself become a
member of the group.

As the group must be self contained, use of the -B group option also asserts the -z
defs option.

-B local
Causes any global symbols, not assigned to a version definition, to be reduced to
local. Version definitions can be supplied via a mapfile and indicate the global
symbols that should remain visible in the generated object. This option achieves the
same symbol reduction as the auto-reduction directive available as part of a mapfile
version definition and may be useful when combining versioned and
non-versioned relocatable objects.

-B reduce
When generating a relocatable object, causes the reduction of symbolic information
defined by any version definitions. Version definitions can be supplied via a mapfile
to indicate the global symbols that should remain visible in the generated object.
When a relocatable object is generated, by default version definitions are only
recorded in the output image. The actual reduction of symbolic information will be
carried out when the object itself is used in the construction of a dynamic
executable or shared object. This option is applied automatically when dynamic
executable or shared object is created.

-B symbolic
In dynamic mode only. When building a shared object, binds references to global
symbols to their definitions, if available, within the object. Normally, references to
global symbols within shared objects are not bound until runtime, even if
definitions are available, so that definitions of the same symbol in an executable or
other shared object can override the object’s own definition. ld will issue warnings
for undefined symbols unless -z defs overrides.

The -B symbolic option is intended for specialized dynamic objects and is not
recommended for general use. To reduce the runtime relocation overhead of an
object, the creation of a version definition is recommended.

-c name
Records the configuration file name for use at runtime. Configuration files may be
employed to alter default search paths, provide a directory cache and provide
alternative object dependencies. See crle(1).
Demangles C++ symbol names displayed in diagnostic messages.

When -dy, the default, is specified, ld uses dynamic linking; when -dn is specified, ld uses static linking. See also -B dynamic|static.

Prints debugging information, as specified by each token, to the standard error. The special token help indicates the full list of tokens available.

Sets the entry point address for the output file to be that of the symbol epsym.

Useful only when building a shared object. Specifies that the symbol table of the shared object is used as an auxiliary filter on the symbol table of the shared object specified by name. Multiple instances of this option are allowed. This option may not be combined with the -F option.

Useful only when building a shared object. Specifies that the symbol table of the shared object is used as a filter on the symbol table of the shared object specified by name. Multiple instances of this option are allowed. This option may not be combined with the -f option.

In dynamic mode only, produces a shared object. Undefined symbols are allowed.

In dynamic mode only, when building a shared object, records name in the object’s dynamic section. name will be recorded in dynamic objects that are linked with this object rather than the object’s file system name. Accordingly, name will be used by the runtime linker as the name of the shared object to search for at runtime.

Ignores LD_LIBRARY_PATH. This option is useful when an LD_LIBRARY_PATH setting is in effect to influence the runtime library search, which would interfere with the link-editing being performed.

When building an executable, uses name as the path name of the interpreter to be written into the program header. The default in static mode is no interpreter; in dynamic mode, the default is the name of the runtime linker, ld. so.1(1). Either case may be overridden by -l name. exec(2) will load this interpreter when it loads a.out and will pass control to the interpreter rather than to a.out directly.

Searches a library libx.so or libx.a, the conventional names for shared object and archive libraries, respectively. In dynamic mode, unless the -B static option is in effect, ld searches each directory specified in the library search path for a libx.so or libx.a file. The directory search stops at the first directory containing either. ld chooses the file ending in .so if -lx expands to two files with names of
ld(1)

the form libx.so and libx.a. If no libx.so is found, then ld accepts libx.a. In static mode, or when the -B static option is in effect, ld selects only the file ending in .a. ld searches a library when it encounters its name, so the placement of -l is significant.

-L path
Add path to the library search directories. ld searches for libraries first in any directories specified by the -L options and then in the standard directories. This option is useful only if it precedes the -l options to which it applies on the command line. The environment variable LD_LIBRARY_PATH may be used to supplement the library search path (see LD_LIBRARY_PATH below).

-m
Produces a memory map or listing of the input/output sections, together with any non-fatal multiply-defined symbols, on the standard output.

-M mapfile
Reads mapfile as a text file of directives to ld. This option may be specified multiple times. If mapfile is a directory, then all regular files, as defined by stat(2), within the directory will be processed. See Linker and Libraries Guide for a description of mapfiles. There are mapfiles in /usr/lib/ld that show the default layout of programs, mapfiles for linking 64–bit programs above or below 4 gigabytes, and a mapfile for establishing a non-executable stack within an application. See the FILES section below.

-N string
This option causes a DT_NEEDED entry to be added to the .dynamic section of the object being built. The value of the DT_NEEDED string will be the string specified on the command line. This option is position dependent, and the DT_NEEDED .dynamic entry will be relative to the other dynamic dependencies discovered on the link-edit line. This option is useful for specifying dependencies within device driver relocatable objects when combined with the -dy and -r options.

-o outfile
Produces an output object file named outfile. The name of the default object file is a.out.

-p auditlib
Identifies an audit library, auditlib, that is used to audit this object at runtime. Any shared object identified as requiring auditing of itself has this requirement inherited by any object specifying this shared object as a dependency (see -P option).

-P auditlib
Identifies an audit library, auditlib, that is used to audit this object’s dependencies at runtime. Dependency auditing can also be inherited from dependencies identified as requiring auditing (see -p option).

-Q y | n
Under -Q y, an ident string is added to the .comment section of the output file to identify the version of the link-editor used to create the file. This results in multiple
ld identifies when there have been multiple linking steps, such as when using `ld -r`. This is identical with the default action of the `cc` command. `-Qn` suppresses version identification.

-`r` Combines relocatable object files to produce one relocatable object file. `ld` will not complain about unresolved references. This option cannot be used with the `-a` option.

-`R path` A colon-separated list of directories used to specify library search directories to the runtime linker. If present and not NULL, it is recorded in the output object file and passed to the runtime linker. Multiple instances of this option are concatenated together with each `path` separated by a colon.

-`s` Strips symbolic information from the output file. Any debugging information, that is `.debug`, `.line`, and `.stab` sections, and their associated relocation entries will be removed. Except for relocatable files or shared objects, the symbol table and string table sections will also be removed from the output object file.

-`S supportlib` The shared object `supportlib` is loaded with the link-editor and given information regarding the linking process. Support shared objects may also be supplied using the `SGS_SUPPORT` environment variable. See Linker and Libraries Guide for more details.

-`t` Turns off the warning for multiply-defined symbols that have different sizes or alignments.

-`u symname` Enters `symname` as an undefined symbol in the symbol table. This is useful for loading entirely from an archive library, since initially the symbol table is empty, and an unresolved reference is needed to force the loading of the first routine. The placement of this option on the command line is significant; it must be placed before the library that will define the symbol.

-`V` Outputs a message giving information about the version of `ld` being used.

-`Y P, dirlist` Changes the default directories used for finding libraries. `dirlist` is a colon-separated path list.

-`z absexec` Useful only when building a dynamic executable. Specifies that references to external absolute symbols should be resolved immediately instead of being left for resolution at runtime. In very specialized circumstances, this option removes text relocations that can result in excessive swap space demands by an executable.
**-z allextract | defaultextract | weakextract**

Alters the extraction criteria of objects from any archives that follow. By default, archive members are extracted to satisfy undefined references and to promote tentative definitions with data definitions. Weak symbol references do not trigger extraction. Under `-z allextract`, all archive members are extracted from the archive. Under `-z weakextract`, weak references trigger archive extraction. `-z defaultextract` provides a means of returning to the default following use of the former extract options.

**-z combrelloc**

Combines multiple relocation sections. Historically, relocation sections are maintained in a one-to-one relationship with the sections to which the relocations will be applied. When building an executable or shared object, `ld` sorts the entries of data relocation sections by their symbol reference so as to reduce runtime symbol lookup. Combining multiple data relocation sections allows optimal sorting and hence the least relocation overhead when objects are loaded into memory.

**-z def | nodefs**

The `-z def` option forces a fatal error if any undefined symbols remain at the end of the link. This is the default when an executable is built, but for historic reasons is not the default when building a shared object. Use of the `-z def` option is recommended, as it assures the object being built is self-contained, that is, that all its symbolic references are resolved internally or to the object’s immediate dependencies.

The `-z nodefs` option allows undefined symbols. For historic reasons, this is the default when a shared object is built. When used with executables, the behavior of references to such undefined symbols is unspecified. Use of the `-z nodefs` option is not recommended.

**-z endfiltee**

Marks a filtee so that when processed by a filter it terminates any further filtee searches by the filter.

**-z finiarray=function**

Appends an entry to the `.finiarray` section of the object being built. If no `.finiarray` section is present, one is created. The new entry is initialized to point to `function`. See Linker and Libraries Guide for more details.

**-z groupperm | nogroupperm**

Assigns, or deassigns each dependency that follows to a unique group. Assigning a dependency to a group has the same effect as if the dependency had been built using the `-B group` option.

**-z ignore | record**

Ignores, or records, dynamic dependencies that are not referenced as part of the link-edit. By default, `-z record` is in effect.
-z initarray=FUNCTION
   Appends an entry to the .initarray section of the object being built. If no
   .initarray section is present, one is created. The new entry is initialized to point
   to FUNCTION. See Linker and Libraries Guide for more details.

- z initfirst
   Marks the object so that its runtime initialization occurs before the runtime
   initialization of any other objects brought into the process at the same time. In
   addition, the object runtime finalization will occur after the runtime finalization of
   any other objects removed from the process at the same time. This option is only
   meaningful when building a shared object.

- z interpose
   Marks the object as an interposer. When direct bindings are in effect (see -B
direct), the runtime linker will search for symbols in any interposers before the
   object associated to the direct binding.

- z lazyload | nolazyload
   Enables or disables the marking of dynamic dependencies to be lazily loaded.
   Dynamic dependencies which are marked lazyload will not be loaded at initial
   process start-up, but instead will be delayed until the first binding to the object is
   made.

- z ld32=arg1,arg2,...
- z ld64=arg1,arg2,...
   The class of the link-editor is affected by the class of the output file being created
   and by the capabilities of the underlying operating system. This option provides a
   means of defining any link-editor argument, such that it will only be interpreted,
   respectively, by the 32– or 64–bit class of the link-editor.

   For example, support libraries are class specific, so the correct class of support
   library can be insured using:
   
   ld ... -z ld32=-Saudit32.so.1 -z ld64=-Saudit64.so.1 ...

   Note: The class of link-editor invoked is in part determined from the ELF class of
   the first input relocatable file seen on the command line. This determination is
   carried out prior to any -z ld[32|64] processing.

- z loadfltr
   Marks the object to require that when building a filter, its filtees be processed
   immediately at runtime. Normally, filter processing is delayed until a symbol
   reference is bound to the filter. The runtime processing of an object that contains
   this flag mimics that which occurs if the LD_LOADFLTR environment variable is in
   effect. See ld.so.1(1).

- z muldefs
   Allows multiple symbol definitions. By default, multiple symbol definitions that
   occur between relocatable objects will result in a fatal error condition. This option
   suppresses the error condition and allows the first symbol definition to be taken.
-z nodelfaultlib
Marks the object so that the runtime default library search path (used after any
LD_LIBRARY_PATH or runpaths) is ignored. This option implies that all
dependencies of the object can be satisfied from its runpath.

- z nodelete
Marks the object as non-deletable at runtime. The runtime processing of an object
that contains this flag mimics that which occurs if the object is added to a process
using dlopen(3DL) with the RTLD_NODELETE mode.

- z nodlopen
Marks the object as not available to dlopen(3DL), either as the object specified by
the dlopen(), or as any form of dependency required by the object specified by
the dlopen(). This option is only meaningful when building a shared object.

- z nodump
Marks the object as not available to dldump(3DL).

- z nopartial
If there are any partially initialized symbols in the input relocatable object files, the
partially initialized symbols are expanded when the output file is generated.

- z noversion
Does not record any versioning sections. Any version sections or associated
.dynainamic section entries will not be generated in the output image.

- z now
Marks the object to override the runtime linker’s default mode and require non-lazy
runtime binding. This is similar to adding the object to the process by using
dlopen(3DL) with the RTLD_NOW mode, or setting the LD_BIND_NOW environment
variable in effect. See ld.so.1(1).

- z origin
Marks the object as requiring immediate $ORIGIN processing at runtime. This
option is only maintained for historic compatibility, as the runtime analysis of
objects to provide for $ORIGIN processing is now default.

- z preinitarray=function
Appends an entry to the .preinitarray section of the object being built. If no
.preinitarray section is present, one is created. The new entry is initialized to
point to function. See Linker and Libraries Guide for more details.

- z redlocsym
Eliminates all local symbols except for the SECT symbols from the symbol table
SHT_SYMTAB. All relocations that refer to local symbols will be updated to refer to
the corresponding SECT symbol.

- z rescan
Rescans the archive files provided to the link-edit. By default, archives are
processed once as they appear on the command line. Archives are traditionally
specified at the end of the command line so that their symbol definitions resolve
any preceding references. However, it is often necessary to specify the archives
multiple times to satisfy their own interdependencies.
The `-z rescan` option causes the entire archive list to be reprocessed in an attempt to locate additional archive members that resolve symbol references. This archive rescanning continues until a pass over the archive list occurs in which no new members are extracted.

`-z text`
In dynamic mode only, forces a fatal error if any relocations against non-writable, allocatable sections remain. For historic reasons, this is not the default when building an executable or shared object. However, its use is recommended to insure that the text segment of the dynamic object being built is shareable between multiple running processes, and that the object incurs the least relocation overhead when loaded into memory.

`-z textoff`
In dynamic mode only, allows relocations against all allocatable sections, including non-writable ones. This is the default when building a shared object.

`-z textwarn`
In dynamic mode only, lists a warning if any relocations against non-writable, allocatable sections remain. This is the default when building an executable.

`-z verbose`
This option provides additional warning diagnostics during a link-edit. Presently, its implementation conveys suspicious use of displacement relocations, but in future it may be enhanced to provide additional diagnostics deemed too noisy to be generated by default.

`LD_LIBRARY_PATH`
A list of directories in which to search for libraries specified with the `-l` option. Multiple directories are separated by a colon. In the most general case, it will contain two directory lists separated by a semicolon:

dirlist1;dirlist2

If `ld` is called with any number of occurrences of `-L`, as in:

```
ld ... -Lpath1 ... -Lpathn ...
```

then the search path ordering is:

dirlist1 path1 ... pathn dirlist2 LIBPATH

When the list of directories does not contain a semicolon, it is interpreted as `dirlist2`.

The `LD_LIBRARY_PATH` environment variable also affects the runtime linkers searching for dynamic dependencies.

This environment variable can be specified with a `_32` or `_64` suffix. This makes the environment variable specific, respectively, to 32-bit or 64-bit processes and overrides any non-suffixed version of the environment variable that may be in effect.
LD_NOEXEC_64
Suppresses the automatic execution of the 64-bit link-editor. By default, the
link-editor will execute its 64-bit version when the ELF class of the first input
relocatable file it reads identifies it as a 64-bit object.

LD_OPTIONS
A default set of options to ld. LD_OPTIONS is interpreted by ld just as though its
value had been placed on the command line, immediately following the name used
to invoke ld, as in:

 ld $LD_OPTIONS ... other-arguments ...

LD_RUN_PATH
An alternative mechanism for specifying a runpath to the link-editor (see -R
option). If both LD_RUN_PATH and the -R option are specified, -R supersedes.

SGS_SUPPORT
Provides a colon-separated list of shared objects that are loaded with the link-editor
and given information regarding the linking process. This environment variable can
be specified with a _32 or _64 suffix. This makes the environment variable specific,
respectively, to the 32-bit or 64-bit class of ld and overrides any non-suffixed
version of the environment variable that may be in effect. See also the -S option.

Notice that environment variable-names beginning with the characters 'LD_' are
reserved for possible future enhancements to ld and ld.so.1(1).

FILES
libx.so
shared object libraries.

libx.a
archive libraries.

a.out
default output file.

LIBPATH
/usr/lib for 32-bit libraries, or /usr/lib/64 for 64-bit libraries.

/usr/lib/ld/map.bssalign
map file providing a template for aligning bss.

/usr/lib/ld/map.default
map file showing default layout of 32-bit programs.

/usr/lib/ld/map.noe stk
map file showing a non-executable stack definition.

/usr/lib/ld/sparcv9/map.default
map file showing default layout of 64-bit SPARC V9 programs.

/usr/lib/ld/sparcv9/map.above4G
map file showing suggested layout above 4 gigabytes of 64-bit SPARC V9 programs.

/usr/lib/ld/sparcv9/map.below4G
map file showing suggested layout below 4 gigabytes of 64-bit SPARC V9 programs.
ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTES TYPE</th>
<th>ATTRIBUTES VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWtoo</td>
</tr>
</tbody>
</table>

SEE ALSO

as(1), crle(1), gprof(1), ld.so.1(1), pvs(1), exec(2), stat(2), diopen(3DL),
dldump(3DL), elf(3ELF), ar(3HEAD), a.out(4), attributes(5)

Linker and Libraries Guide

NOTES

Default options applied by ld are maintained for historic reasons. In today’s
programming environment, where dynamic objects dominate, alternative defaults
would often make more sense. However, historic defaults must be maintained to
insure compatibility with existing program development environments. Historic
defaults are called out wherever possible in this manual. For a description of current
recommended options, see the “Link-Editor Quick Reference” in the Linker and
Libraries Guide.

If the file being created by ld already exists, it will be truncated after all input files
have been processed and overridden with the new file contents. ld does not create a
temporary file as part of the link-edit, since multiple instances of large output files
frequently exhaust system resources. The drawback of overriding an existing file
occurs if the file is in use by a running process. In this case, the process may be
prematurely terminated as the output files image is created. This situation can be
avoided by removing the output file before performing the link-edit. This removal is
not detrimental to the running process, as it frees up the file system namespace, not
the actual disk space, for the new output file creation. The disk space of a removed file
is freed when the last process referencing the file terminates.
NAME | ld – link editor, dynamic link editor

SYNOPSIS | /usr/ucb/ld [options]

DESCRIPTION | /usr/ucb/ld is the link editor for the BSD Compatibility Package. /usr/ucb/ld is identical to /usr/ccs/bin/ld (see ld(1)) except that BSD libraries and routines are included before the base libraries and routines.

OPTIONS | /usr/ucb/ld accepts the same options as /usr/ccs/bin/ld, with the following exceptions:

-L dir | Add dir to the list of directories searched for libraries by /usr/ccs/bin/ld. Directories specified with this option are searched before /usr/ucb/lib and /usr/lib.

-Y LU,dir | Change the default directory used for finding libraries. Warning: This option may have unexpected results, and should not be used.

FILES | /usr/lib

/usr/lib/libx.a

/usr/ucb/lib

/usr/ucblib

/usr/ucblib/libx.a

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

SEE ALSO | ar(1), as(1), cc(1B), ld(1), lorder(1), strip(1), tsort(1), attributes(5)
lda – LDAP as a naming repository

LDAP refers to Lightweight Directory Access Protocol, which is an industry standard for accessing directory servers. By initializing the client using `ldapclient(1M)` and using the keyword `ldap` in the name service switch file, `/etc/nsswitch.conf`, Solaris clients can obtain naming information from an LDAP server. Information such as usernames, hostnames, and passwords are stored on the LDAP server in a Directory Information Tree or DIT. The DIT consists of entries which in turn are composed of attributes. Each attribute has a type and one or more values.

Solaris LDAP clients use the LDAP v3 protocol to access naming information from LDAP servers. The LDAP server must support the object classes and attributes defined in RFC2307bis (draft), which maps the naming service model on to LDAP. As an alternate to using the schema defined in RFC2307bis (draft), the system can be configured to use other schema sets and the schema mapping feature is configured to map between the two. Refer to the System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP) for more details.

The `ldapclient(1M)` utility can make a Solaris machine an LDAP client by setting up the appropriate directories, files, and configuration information. The LDAP client caches this configuration information in local cache files. This configuration information is accessed through the `ldap_cachemgr(1M)` daemon. This daemon also refreshes the information in the configuration files from the LDAP server, providing better performance and security. The `ldap_cachemgr` must run at all times for the proper operation of the naming services.

There are two types of configuration information, the information available through a profile, and the information configured per client. The profile contains all the information as to how the client accesses the directory. The credential information for proxy user is configured on a per client basis and is not downloaded through the profile.

The profile contains server-specific parameters that are required by all clients to locate the servers for the desired LDAP domain. This information could be the server’s IP address and the search base Distinguished Name (DN), for instance. It is configured on the client from the default profile during client initialization and is periodically updated by the `ldap_cachemgr` daemon when the expiration time has elapsed.

Client profiles can be stored on the LDAP server and may be used by the `ldapclient` utility to initialize an LDAP client. Using the client profile is the easiest way to configure a client machine. See `ldapclient(1M)`.

Credential information includes client-specific parameters that are used by a client. This information could be the Bind DN (LDAP “login” name) of the client and the password. If these parameters are required, they are manually defined during the initialization through `ldapclient(1M)`.

The naming information is stored in containers on the LDAP server. A container is a non-leaf entry in the DIT that contains naming service information. Containers are similar to maps in NIS and tables in NIS+. A default mapping between the NIS
databases and the containers in LDAP is presented below. The location of these containers as well as their names can be overridden through the use of serviceSearchDescriptors. For more information see ldapclient(1M).

<table>
<thead>
<tr>
<th>Database</th>
<th>Object Class</th>
<th>Container</th>
</tr>
</thead>
<tbody>
<tr>
<td>passwd</td>
<td>posixAccount</td>
<td>ou=people,dc=...</td>
</tr>
<tr>
<td></td>
<td>shadowAccount</td>
<td></td>
</tr>
<tr>
<td>group</td>
<td>posixGroup</td>
<td>ou=Group,dc=...</td>
</tr>
<tr>
<td>services</td>
<td>ipService</td>
<td>ou=Services,dc=</td>
</tr>
<tr>
<td>protocols</td>
<td>ipProtocol</td>
<td>ou=Protocols,dc=</td>
</tr>
<tr>
<td>rpc</td>
<td>oncRpc</td>
<td>ou=Rpc,dc=...</td>
</tr>
<tr>
<td>hosts</td>
<td>ipHost</td>
<td>ou=Hosts,dc=...</td>
</tr>
<tr>
<td>ipnodes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ethers</td>
<td>ieee802Device</td>
<td>ou=Ethers,dc=...</td>
</tr>
<tr>
<td>bootparams</td>
<td>bootableDevice</td>
<td>ou=Ethers,dc=...</td>
</tr>
<tr>
<td>networks</td>
<td>ipNetwork</td>
<td>ou=Networks,dc=</td>
</tr>
<tr>
<td>netmasks</td>
<td>ipNetwork</td>
<td>ou=Networks,dc=</td>
</tr>
<tr>
<td>netgroup</td>
<td>nisNetgroup</td>
<td>ou=NisNetgroup,dc=</td>
</tr>
<tr>
<td>aliases</td>
<td>mailGroup</td>
<td>ou=Aliases,dc=...</td>
</tr>
<tr>
<td>publickey</td>
<td>nisKeyObject</td>
<td></td>
</tr>
<tr>
<td>generic</td>
<td>nisObject</td>
<td>nisMapName=...,dc=</td>
</tr>
<tr>
<td>printers</td>
<td>printerService</td>
<td>ou=Printers,dc=...</td>
</tr>
<tr>
<td>auth_attr</td>
<td>SolarisAuthAttr</td>
<td>ou=SolarisAuthAttr,dc=</td>
</tr>
<tr>
<td>prof_attr</td>
<td>SolarisProfAttr</td>
<td>ou=SolarisProfAttr,dc=</td>
</tr>
<tr>
<td>exec_attr</td>
<td>SolarisExecAttr</td>
<td>ou=SolarisExecAttr,dc=</td>
</tr>
<tr>
<td>user_attr</td>
<td>SolarisUserAttr</td>
<td>ou=people,dc=...</td>
</tr>
<tr>
<td>audit_attr</td>
<td>SolarisAuditAttr</td>
<td>ou=people,dc=...</td>
</tr>
</tbody>
</table>

The security model for clients is defined by a combination of the credential level to be used, the authentication method, and the PAM module to be used, that is, pam_unix versus pam_ldap. The credential level defines what credentials the client should use to authenticate to the directory server, and the authentication method defines the method of choice. Both these can be set with multiple values. The Solaris LDAP supports the following values for credential level:
The Solaris LDAP supports the following values for authentication method:

- none
- simple
- sasl/CRAM-MD5
- sasl/DIGEST-MD5
- tls:simple
- tls:sasl/CRAM-MD5
- tls:sasl/DIGEST-MD5

More protection can be provided by means of access control, allowing the server to grant access for certain containers or entries. Access control is specified by Access Control Lists (ACL’s) that are defined and stored in the LDAP server. The Access Control Lists on the LDAP server are called Access Control Instructions (ACI’s) by the iPlanet Directory Server. Each ACL or ACI specifies one or more directory objects, for example, the cn attribute in a specific container, one or more clients to whom you grant or deny access, and one or more access rights that determine what the clients can do to or with the objects. Clients can be users or applications. Access rights can be specified as read and write, for example. Refer to the System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP) regarding the restrictions on ACL’s and ACI’s when using LDAP as a naming repository.

A sample nsswitch.conf(4) file called nsswitch.ldap is provided in the /etc directory. This is copied to /etc/nsswitch.conf by the ldapclient(1M) utility. This file uses LDAP as a repository for the different databases in the nsswitch.conf file.

The following is a list of the user commands related to LDAP:

- idsconfig(1M) Prepares an iPlanet Directory Server to be ready to support Solaris LDAP clients.
- ldapaddent(1M) Create LDAP entries from corresponding /etc files.
- ldapclient(1M) Initialize LDAP clients, or generate a configuration profile to be stored in the directory.
- ldaplist(1) List the contents of the LDAP naming space.

FILES

- /var/ldap/ldap_client_cred
- /var/ldap/ldap_client_file
- /etc/nsswitch.conf

Files that contain the LDAP configuration of the client. Do not manually modify these files. Their content is not guaranteed to be human readable. Use ldapclient(1M) to update them.

Configuration file for the name-service switch.
Sample configuration file for the name-service switch configured with LDAP and files.

PAM framework configuration file.

SEE ALSO
ldaplist(1), idsconfig(1M), ldap_cachemgr(1M), ldapaddent(1M),
ldapclient(1M), nsswitch.conf(4), pam.conf(4), pam_authtok_check(5),
pam_authtok_get(5), pam_authtok_store(5), pam_dhkeys(5), pam_ldap(5),
pam_passwd_auth(5), pam_unix(5), pam_unix_account(5), pam_unix_auth(5),
pam_unix_session(5),

NOTES
The pam_unix(5) module might not be supported in a future release. Similar
functionality is provided by pam_authtok_check(5), pam_authtok_get(5),
pam_authtok_store(5), pam_dhkeys(5), pam_passwd_auth(5),
pam_unix_account(5), pam_unix_auth(5), and pam_unix_session(5).
ldapdelete – ldap delete entry tool

SYNOPSIS

           [-w passwd] [-h ldaphost] [-M authentication] [-p ldapport] [dn...]

DESCRIPTION

The ldapdelete utility opens a connection to an LDAP server, then binds and
deletes one or more entries. If one or more dn arguments are provided, entries with
those distinguished names are deleted. If no dn arguments are provided, a list of DNs
is read from file, if the -f option is specified, or from standard input.

OPTIONS

The following options are supported:

- Continuous operation mode. Errors are reported, but
  ldapdelete will continue with deletions. The default
  is to exit after reporting an error.
  -c

- Sets the LDAP debugging level. Useful levels of
depending on ldapdelete are:
  -d debuglevel

  1 Trace
  2 Packets
  4 Arguments
  32 Filters
  128 Access control

  To request more than one category of debugging
  information, add the masks. For example, to request
  trace and filter information, specify a debuglevel of 33.

- Uses the distinguished name binddn to bind to the
directory.
  -D binddn

- Reads the entry deletion information from file instead
  of from standard input.
  -f file

- Specifies an alternate host on which the slapd server is
  running.
  -h ldaphost

- Specifies the authentication mechanism used to bind to
  the directory.
  -M authentication

  The default authentication method for ldapdelete is
  simple bind. Simple bind sends the password to the
  server in the clear. The password is subject to snooping
  if the server is not local. You must use special care
  when you use this command with the default
  authentication method. If your server supports the
challenge response method CRAM-MD5 authentication method, you can override the default authentication method by using the -M option with CRAM-MD5 as the value for authentication.

The bind DN and bind password are mandatory with this option.

- **n**
  Shows what would be done, but does not actually delete entries. Useful in conjunction with options -v and -d for debugging.

- **o**
  Specifies the security layer for the mechanism (none, integrity, privacy).

- **p ldapport**
  Specifies an alternate TCP port where the slapd server is listening.

- **v**
  Uses verbose mode, with diagnostics written to standard output.

- **w passwd**
  Use passwd as the password for authentication to the directory. When you use -w passwd to specify the password to be used for authentication, the password is visible to other users of the system by means of the ps command, in script files or in shell history. If you use the ldapdelete command without this option, the command will prompt for the password and read it from standard in. When used without the -w option, the password will not be visible to other users.

**OPERANDS**
The following operand is supported:

- **dn**
  Specifies one or several distinguished names of entries to delete.

**EXAMPLES**

**EXAMPLE 1** Deleting an entry

To delete the entry named with commonName Delete Me directly below the XYZ Corporation organizational entry, use the following command:

```bash
example% ldapdelete -D "cn=Administrator, o=XYZ, c=US" "cn=Delete Me, o=XYZ, c=US"
```

**ATTRIBUTES**

See attributes(5) for a description of the following attributes:

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcstu</td>
</tr>
<tr>
<td>Stability Level</td>
<td>Evolving</td>
</tr>
</tbody>
</table>
**EXIT STATUS**

The following exit values are returned:

<table>
<thead>
<tr>
<th>Exit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>Non-zero</td>
<td>An error occurred. A diagnostic message is written to standard error.</td>
</tr>
</tbody>
</table>

**SEE ALSO**

ldapadd(1), ldapmodify(1), ldapmodrdn(1), ldapsearch(1),
ldap_get_option(3LDAP), ldap_set_option(3LDAP), attributes(5)
ldaplist(1)

NAME
ldaplist – search and list naming information from a LDAP directory using the configured profile

SYNOPSIS
/usr/bin/ldaplist [-dlv] [database [key] ...]
/usr/bin/ldaplist -h

DESCRIPTION
The ldaplist utility searches for and lists the naming information from the LDAP directory service defined in the LDAP configuration files generated by ldapclient(1M) during the client initialization phase. The Solaris LDAP client must be set up in order to use this utility.

The database is either a container name or a database name as defined in nsswitch.conf(4). A container is a non-leaf entry in the Directory Information Tree (DIT) that contains naming service information. The container name is the LDAP Relative Distinguished Name (RDN) of the container relative to the defaultSearchBase as defined in the configuration files. For example, for a container named ou=people, the database name is the database specified in nsswitch.conf. This database is mapped to a container, for example, passwd maps to ou=people. If an invalid database is specified, it will be mapped to a generic container, for example, nisMapName=name).

The key is the attribute value to be searched in the database. You can specify more than one key to be searched in the same database. The key can be specified in either of two forms: attribute=value or value. In the first case, ldaplist passes the search key to the server. In the latter case, an attribute is assigned depending on how the database is specified. If the database is a container name, then the "cn" attribute type is used. If it is a valid database name as defined in the nsswitch.conf, then a predefined attribute type is used (see table below). If it is an invalid database name, then cn is used as the attribute type.

The ldaplist utility relies on the Schema defined in the RFC 2307bis, currently an IETF draft. The data stored on the LDAP server must be stored based on this Schema, unless the profile contains schema mapping definitions. For more information on schema mapping see ldapclient(1M). The following table lists the default mapping from the database names to the container, the LDAP object class, and the attribute type used if not defined in the key.

<table>
<thead>
<tr>
<th>Database</th>
<th>Object Class</th>
<th>Attribute Type</th>
<th>Container</th>
</tr>
</thead>
<tbody>
<tr>
<td>aliases</td>
<td>mailGroup</td>
<td>cn</td>
<td>ou=Aliases</td>
</tr>
<tr>
<td>automount</td>
<td>nisObject</td>
<td>cn</td>
<td>nisMapName=auto_*</td>
</tr>
<tr>
<td>bootparams</td>
<td>bootableDevice</td>
<td>cn</td>
<td>ou=Ethers</td>
</tr>
<tr>
<td>ethers</td>
<td>ieee802Device</td>
<td>cn</td>
<td>ou=Ethers</td>
</tr>
<tr>
<td>group</td>
<td>posixgroup</td>
<td>cn</td>
<td>ou=Group</td>
</tr>
<tr>
<td>Database</td>
<td>Object Class</td>
<td>Attribute Type</td>
<td>Container</td>
</tr>
<tr>
<td>------------</td>
<td>--------------</td>
<td>----------------</td>
<td>------------</td>
</tr>
<tr>
<td>hosts</td>
<td>ipHost</td>
<td>cn</td>
<td>ou=Hosts</td>
</tr>
<tr>
<td>ipnodes</td>
<td>ipHost</td>
<td>cn</td>
<td>ou=Hosts</td>
</tr>
<tr>
<td>netgroup</td>
<td>ipNetgroup</td>
<td>cn</td>
<td>ou=Netgroup</td>
</tr>
<tr>
<td>netmasks</td>
<td>ipNetwork</td>
<td>ipnetworknumber</td>
<td>ou=Networks</td>
</tr>
<tr>
<td>networks</td>
<td>ipNetwork</td>
<td>ipnetworknumber</td>
<td>ou=Networks</td>
</tr>
<tr>
<td>passwd</td>
<td>posixAccount</td>
<td>uid</td>
<td>ou=People</td>
</tr>
<tr>
<td>protocols</td>
<td>ipProtocol</td>
<td>cn</td>
<td>ou=Protocols</td>
</tr>
<tr>
<td>publickey</td>
<td>nisKeyObject</td>
<td>uidnumber</td>
<td>ou=People</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cn</td>
<td>ou=Hosts</td>
</tr>
<tr>
<td>rpc</td>
<td>oncRpc</td>
<td>cn</td>
<td>ou=Rpc</td>
</tr>
<tr>
<td>services</td>
<td>ipService</td>
<td>cn</td>
<td>ou=Services</td>
</tr>
<tr>
<td>printers</td>
<td>printerService</td>
<td>printer-uri</td>
<td>ou=printers</td>
</tr>
<tr>
<td>auth_attr</td>
<td>SolarisAuthAttr</td>
<td>nameT</td>
<td>ou=SolarisAuthAttr</td>
</tr>
<tr>
<td>prof_attr</td>
<td>SolarisProfAttr</td>
<td>nameT</td>
<td>ou=SolarisProfAttr</td>
</tr>
<tr>
<td>exec_attr</td>
<td>SolarisExecAttr</td>
<td>nameT</td>
<td>ou=SolarisProfAttr</td>
</tr>
<tr>
<td>user_attr</td>
<td>SolarisUserAttr</td>
<td>uidT</td>
<td>ou=people</td>
</tr>
<tr>
<td>audit_user</td>
<td>SolarisAuditUser</td>
<td>uidT</td>
<td>ou=people</td>
</tr>
</tbody>
</table>

- auto_* represents auto_home, auto_direct, ...
- If the key starts with a digit, it will be interpreted as an uid number.
- If the key starts with a non-digit, it will be interpreted as a host name.

The ldaplist utility supports substring search by using the wildcard "*" in the key. For example, "my*" will match any strings that starts with "my". In some shell environments, keys containing the wildcard may need to be quoted.

If the key is not specified, all the containers in the current search baseDN will be listed.

**OPTIONS**

The following options are supported:

- `-d` Lists the attributes for the specified database, rather than the entries. By default, the entries are listed.
- `-h` Lists the database mapping.
- `-l` Lists all the attributes for each entry matching the search criteria. By default, ldaplist lists only the Distinguished Name of the entries found.
Sets verbose mode. The ldaplist utility will also print the filter used to search for the entry. The filter will be prefixed with "+++".

EXAMPLE 1 Listing All Entries in the Hosts Database
example$ ldaplist hosts

EXAMPLE 2 Listing All Entries in a Non-Standard Database ou=new
example$ ldaplist ou=new

EXAMPLE 3 Finding “user1” in the passwd Database
example$ ldaplist passwd user1

EXAMPLE 4 Finding the Entry With Service Port of 4045 in the services Database
example$ ldaplist services ipServicePort=4045

EXAMPLE 5 Finding All Users With Username Starting with new in the passwd Database
example$ ldaplist passwd ‘new*’

EXAMPLE 6 Listing the Attributes for the hosts Database
example$ ldaplist -d hosts

EXIT STATUS
The following exit values are returned:
0 Successfully matched some entries.
1 Successfully searched the table and no matches were found.
2 An error occurred. An error message is output.

FILES
/var/ldap/ldap_client_file
/var/ldap/ldap_client_cred Files that contain the LDAP configuration of the client. Do not manually modify these files. Their content is not guaranteed to be human readable. To update these files, use ldapclient(1M)

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWnisu</td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Evolving</td>
</tr>
</tbody>
</table>
SEE ALSO

1dap(1), 1dapadd(1), 1dapdelete(1), 1dapmodify(1), 1dapmodrdn(1),
1dapsearch(1), idsconfig(1M), ldap_cachemgr(1M), ldapaddent(1M),
ldapclient(1M), suninstall(1M), resolv.conf(4), attributes(5)

NOTES

RFC 2307bis is an IETF informational document in draft stage that defines an approach
for using LDAP as a naming service.
ldapmodify(1)

NAME
ldapmodify, ldapadd – ldap entry addition and modification tools

SYNOPSIS
[[-l nb-ldap-connections]]

[-h ldaphost] [-p ldapport] [-f file] [[-l nb-ldap-connections]]

DESCRIPTION
The ldapmodify utility opens a connection to an LDAP server, binds and modifies or
adds entries. The entry information is read from standard input or from file, specified
using the -f option. The ldapadd utility is implemented as a hard link to the
ldapmodify tool. When invoked as ldapadd, the -a (add new entry) option is
turned on automatically.

Both ldapadd and ldapmodify reject duplicate attribute-name/value pairs for the
same entry.

OPTIONS
The following options are supported:

-a
Adds new entries. The default for ldapmodify is to
modify existing entries. If invoked as ldapadd, this
option is always set.

-c
Specifies continuous operation mode. Errors are
reported, but ldapmodify and ldapadd continue
with modifications. The default is to exit after reporting
an error.

-D binddn
Uses the distinguished name binddn to bind to the
directory.

-d debuglevel
Sets the LDAP debugging level. Useful levels of
debugging for ldapmodify and ldapadd are:

1 Trace
2 Packets
4 Arguments
32 Filters
128 Access control

To request more than one category of debugging
information, add the masks. For example, to request
trace and filter information, specify a debuglevel of 33.

-F
Forces application of all changes regardless of the
content of input lines that begin with replica: . By
default, replica: lines are compared against the
ldapmodify(1)

LDAP server host and port in use to decide whether a relog record should be applied.

-f file
Reads the entry modification information from file instead of from standard input.

-h ldaphost
Specifies an alternate host on which the slapd server is running.

-l nb-ldap-connections
Specifies the number of LDAP connections that ldapadd or ldapmodify will open to process the modifications in the directory. The default is one connection.

-M authentication
Specifies the authentication mechanism used to bind to the directory.

The default authentication method for ldapmodify and ldapadd is simple bind. simple bind sends the password to the server in the clear. The password is subject to snooping if the server is not local. You must use special care when you use this command with the default authentication method. If your server supports the challenge response method CRAM-MD5 authentication method, you can override the default authentication method by using the -M option with CRAM-MD5 as the value for authentication.

The bind DN and bind password are mandatory with this option.

-n
Previews modifications, but makes no changes to entries. Useful in conjunction with -v and -d for debugging.

-p ldapport
Specifies an alternate TCP port where the slapd server is listening.

-r
Replaces existing value with the specified value. This is the default for ldapmodify. When ldapadd is called, or if the -a option is specified, the -r option is ignored.

-v
Uses verbose mode, with diagnostics written to standard output.

-w passwd
Use passwd as the password for authentication to the directory. When you use -w passwd to specify the password to be used for authentication, the password is visible to other users of the system by means of the ps command, in script files or in shell history. If you use either the ldapmodify command or the ldapadd
command without this option, the command will prompt for the password and read it from standard in. When used without the \texttt{-w} option, the password will not be visible to other users.

\textbf{EXAMPLES}

The format of the content of file (or standard input if no \texttt{-f} option is specified) is illustrated in the following examples.

\textbf{EXAMPLE 1} Modifying an entry

The file /tmp/entrymods contains the following modification instructions:

```
   dn: cn=Modify Me, o=XYZ, c=US
   changetype: modify
   replace: mail
   mail: modme@atlanta.xyz.com
   -
   add: title
   title: System Manager
   -
   add: jpegPhoto
   jpegPhoto:< file:///tmp/modme.jpeg
   -
   delete: description
   -
```

The command:

```
exampex% ldapmodify -r -f /tmp/entrymods
```

modifies the Modify Me entry as follows:

1. The current value of the \texttt{mail} attribute is replaced with the value, modme@atlanta.xyz.com.
2. A \texttt{title} attribute with the value, System Manager, is added.
3. A \texttt{jpegPhoto} attribute is added, using the contents of the file, /tmp/modme.jpeg, as the attribute value.
4. The \texttt{description} attribute is removed.

\textbf{EXAMPLE 2} Creating a new entry

The file, /tmp/newentry, contains the following information for creating a new entry:

```
   dn: cn=Ann Jones, o=XYZ, c=US
   objectClass: person
   cn: Ann Jones
   cn: Annie Jones
   sn: Jones
   title: Director of Research and Development
   mail: ajones@londonrd.xyz.us.com
   uid: ajones
```

\textbf{662} man pages section 1: User Commands • Last Revised 30 Jan 2002
EXAMPLE 2 Creating a new entry  (Continued)

The command

```bash
example$ ldapadd -f /tmp/newentry
```

adds a new entry for Ann Jones, using the information in the file.

EXAMPLE 3 Deleting an entry

The file, `/tmp/badentry`, contains the following information about an entry to be deleted:

```
  dn: cn=Ann Jones, o=XYZ, c=US
  changetype: delete
```

The command:

```bash
example$ ldapmodify -f /tmp/badentry
```

removes Ann Jones’ entry.

ATTRIBUTES  See attributes(5) for a description of the following attributes:

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>Stability Level</td>
<td>Evolving</td>
</tr>
</tbody>
</table>

EXIT STATUS  The following exit values are returned:

- `0`  Successful completion.
- Non-zero  An error occurred. A diagnostic message is written to standard error.

SEE ALSO  `ldapdelete(1)`, `ldapmodrdn(1)`, `ldapsearch(1)`, `ldap_get_option(3LDAP)`, `ldap_set_option(3LDAP)`, `attributes(5)`
ldapmodrdn(1)

NAME
ldapmodrdn – ldap modify entry RDN tool

SYNOPSIS
[-h ldaphost] [-M authentication] [-p ldapport] [-f file] [dn rdn]

DESCRIPTION
ldapmodrdn opens a connection to an LDAP server, binds, and modifies the RDN of
entries. The entry information is read from standard input, from file through the use
of the -f option, or from the command-line pair dn and rdn.

OPTIONS
- c              Continuous operation mode. Errors are reported, but
                 ldapmodify continues with modifications. The default
                 is to exit after reporting an error.

- D binddn       Use the distinguished name binddn to bind to the
directory.

- d debuglevel   Set the LDAP debugging level. Useful values of
debuglevel for ldapmodrdn are:

  1       Trace
  2       Packets
  4       Arguments
 32      Filters
 128     Access control

To request more than one category of debugging
information, add the masks. For example, to request
trace and filter information, specify a debuglevel of 33.

- f file         Read the entry modification information from file
                 instead of from standard input or the command-line.

- h ldaphost     Specify an alternate host on which the slapd server is
                 running.

- M authentication
                 Specifies the authentication mechanism used to bind to
                 the directory.

The default authentication method for ldapmodrdn is
simple bind. simple bind sends the password to
the server in the clear. The password is subject to snooping
if the server is not local. You must use special care
when you use this command with the default
authentication method. If your server supports the
challenge response method CRAM-MD5 authentication
method, you can override the default authentication
method by using the -M option with CRAM-MD5 as the
value for authentication.
The bind DN and bind password are mandatory with this option.

-n Show what would be done, but don't actually change entries. Useful in conjunction with -v for debugging.

-p ldapport Specify an alternate TCP port where the slapd server is listening.

-r Remove old RDN values from the entry. By default, old values are kept.

-v Use verbose mode, with diagnostics written to standard output.

-w passwd Use passwd as the password for authentication to the directory. When you use -w passwd to specify the password to be used for authentication, the password is visible to other users of the system by means of the ps command, in script files or in shell history. If you use the ldapmodrdn command without this option, the command will prompt for the password and read it from standard in. When used without the -w option, the password will not be visible to other users.

If the command-line arguments dn and rdn are given, rdn replaces the RDN of the entry specified by the DN, dn.

Otherwise, the contents of file (or standard input if the -f option is not specified) must consist of one or more pair of lines:

    Distinguished Name (DN)
    Relative Distinguished Name (RDN)

Use one or more blank lines to separate each DN/RDN pair.

The file /tmp/entrymods contains:

    cn=Modify Me, o=XYZ, c=US
    cn=The New Me

The command:

    example$ ldapmodify -r -f /tmp/entrymods

changes the RDN of the "Modify Me" entry from "Modify Me" to "The New Me" and the old cn, "Modify Me" is removed.

See attributes(5) for a description of the following attributes:
ldapmodrdn(1)

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>Stability Level</td>
<td>Evolving</td>
</tr>
</tbody>
</table>

SEE ALSO

ldapadd(1), ldapdelete(1), ldapmodify(1), ldapsearch(1), attributes(5)

DIAGNOSTICS

Exit status is 0 if no errors occur. Errors result in a non-zero exit status and a diagnostic message being written to standard error.
ldapsearch

ldapsearch opens a connection to an LDAP server, binds, and performs a search using the filter filter.

If ldapsearch finds one or more entries, the attributes specified by attrs are retrieved and the entries and values are printed to standard output. If no attrs are listed, all attributes are returned.

If one or more entries are found, each entry is written to standard output in the form:

Distinguished Name (DN)
User Friendly Name (if the -u option is used)
attributename=value
attributename=value
attributename=value
...

Multiple entries are separated with a single blank line. If the -F option is used to specify a different separator character, this character will be used instead of the '=' character. If the -t option is used, the name of a temporary file is returned in place of the actual value. If the -A option is given, only the "attributename" is returned and not the attribute value.

OPTIONS

-A
Retrieve attributes only (no values). This is useful when you just want to see whether an attribute is present in an entry and are not interested in the specific value.

-a deref
Specify how aliases dereferencing is done. The possible values for deref are never, always, search, or find to specify respectively that aliases are never dereferenced, always dereferenced, dereferenced when searching, or dereferenced only when finding the base object for the search. The default is to never dereference aliases.

-B
Do not suppress display of non-ASCII values. This is useful when dealing with values that appear in alternate character sets such as ISO-8859.1. This option is automatically set by the -L option.

-b searchbase
Use searchbase as the starting point for the search instead of the default.

-D binddn
Use the distinguished name binddn to bind to the directory.
ldapsearch(1)

-\texttt{d debuglevel} Set the LDAP debugging level. Useful levels of debugging for \texttt{ldapsearch} are:

\begin{itemize}
  \item \texttt{1 Trace}
  \item \texttt{2 Packets}
  \item \texttt{4 Arguments}
  \item \texttt{32 Filters}
  \item \texttt{128 Access control}
\end{itemize}

To request more than one category of debugging information, add the masks. For example, to request trace and filter information, specify a debuglevel of \texttt{33}.

-\texttt{F sep} Use \texttt{sep} as the field separator between attribute names and values. The default separator is \texttt{='}=\texttt{'.} If \texttt{-L} option has been specified, this option is ignored.

-\texttt{f file} Read a series of lines from \texttt{file}, performing one LDAP search for each line. In this case, the \texttt{filter} given on the command line is treated as a pattern where the first occurrence of \texttt{%s} is replaced with a line from \texttt{file}. If \texttt{file} is a single - character, then the lines are read from standard input.

-\texttt{h ldaphost} Specify an alternate host on which the slapd server is running.

-\texttt{L} Display search results in a modified format. This option also turns on the \texttt{-B} option, and causes the \texttt{-F} option to be ignored.

-\texttt{l timelimit} Wait at most \texttt{timelimit} seconds for a search to complete.

-\texttt{M authentication} Specifies the authentication mechanism used to bind to the directory.

The default authentication method for \texttt{ldapsearch} is \texttt{simple bind}. \texttt{Simple bind} sends the password to the server in the clear. The password is subject to snooping if the server is not local. You must use special care when you use this command with the default authentication method. If your server supports the challenge response method \texttt{CRAM-MD5} authentication method, you can override the default authentication method by using the \texttt{-M} option with \texttt{CRAM-MD5} as the value for \texttt{authentication}.

The bind DN and bind password are mandatory with this option.
### ldapsearch(1)

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-n</code></td>
<td>Show what would be done, but do not actually perform the search. Useful in conjunction with <code>-v</code> and <code>-d</code> for debugging.</td>
</tr>
<tr>
<td><code>-p ldapport</code></td>
<td>Specify an alternate TCP port where the slapd server is listening.</td>
</tr>
<tr>
<td><code>-R</code></td>
<td>Do not automatically follow referrals returned while searching.</td>
</tr>
<tr>
<td><code>-s scope</code></td>
<td>Specify the scope of the search. The possible values of <code>scope</code> are <code>base</code>, <code>one</code>, or <code>sub</code> to specify respectively a base object, one-level, or subtree search. The default is <code>sub</code>.</td>
</tr>
<tr>
<td><code>-t</code></td>
<td>Write retrieved values to a set of temporary files. This is useful for dealing with non-ASCII values such as jpegPhoto or audio.</td>
</tr>
<tr>
<td><code>-u</code></td>
<td>Include the user-friendly form of the Distinguished Name (DN) in the output.</td>
</tr>
<tr>
<td><code>-v</code></td>
<td>Run in verbose mode, with diagnostics written to standard output.</td>
</tr>
<tr>
<td><code>-w passwd</code></td>
<td>Use <code>passwd</code> as the password for authentication to the directory. When you use <code>-w passwd</code> to specify the password to be used for authentication, the password is visible to other users of the system by means of the <code>ps</code> command, in script files or in shell history. If you use the <code>ldapsearch</code> command without this option, the command will prompt for the password and read it from standard in. When used without the <code>-w</code> option, the password will not be visible to other users.</td>
</tr>
<tr>
<td><code>-z sizelimit</code></td>
<td>Retrieve at most <code>sizelimit</code> entries for a search to complete.</td>
</tr>
</tbody>
</table>

### EXAMPLES

**EXAMPLE 1 Performing a Subtree Search**

The following command performs a subtree search (using the default search base) for entries with a commonName of "mark smith". The commonName and telephoneNumber values will be retrieved and printed to standard output.

```bash
example% ldapsearch "cn=mark smith" cn telephoneNumber
```

The output looks something like this:

```plaintext
cn=Mark D Smith, ou=Sales, ou=Atlanta, ou=People, o=XYZ, c=US
cn=Mark Smith
cn=Mark David Smith
cn=Mark D Smith 1
cn=Mark D Smith
```
EXAMPLE 1 Performing a Subtree Search  (Continued)

telephoneNumber=+1 123 456-7890

cn=Mark C Smith, ou=Distribution, ou=Atlanta, ou=People, o=XYZ, c=US

cn=Mark Smith

cn=Mark C Smith 1

cn=Mark C Smith

telephoneNumber=+1 123 456-9999

EXAMPLE 2 Performing a Subtree Search Using the Default Search Base

The following command performs a subtree search using the default search base for entries with user id of "mcs". The user-friendly form of the entry’s DN will be output after the line that contains the DN itself, and the jpegPhoto and audio values will be retrieved and written to temporary files.

```
example% ldapsearch -u -t "uid=mcs" jpegPhoto audio
```

The output might look like this if one entry with one value for each of the requested attributes is found:

```
cn=Mark C Smith, ou=Distribution, ou=Atlanta, ou=People, o=XYZ, c=US
Mark C Smith, Distribution, Atlanta, People, XYZ, US
audio=/tmp/ldapsearch-audio-a19924
jpegPhoto=/tmp/ldapsearch-jpegPhoto-a19924
```

EXAMPLE 3 Performing a One Level Search

The following command performs a one-level search at the c=US level for all organizations whose organizationName begins with XY.

```
example% ldapsearch -L -s one -b "c=US" "o=XY*" o description
```

Search results are displayed in the LDIF format. The organizationName and description attribute values will be retrieved and printed to standard output, resulting in output similar to this:

```
dn: o=XYZ, c=US
 o: XYZ
description: XYZ Corporation
```
```
dn: o="XY Trading Company", c=US
 o: XY Trading Company
description: Import and export specialists
```
```
dn: o=XYInternational, c=US
 o: XYInternational
 o: XYI
 o: XY International
```

EXIT STATUS  The following exit values are returned:

- 0  Successful completion.
An error occurred. A diagnostic message is written to standard error.

**ATTRIBUTES**

See attributes(5) for a description of the following attributes:

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>Stability Level</td>
<td>Evolving</td>
</tr>
</tbody>
</table>

**SEE ALSO**

ldapadd(1), ldapdelete(1), ldapmodify(1), ldapmodrdn(1), attributes(5)
The `ldd` utility lists the dynamic dependencies of executable files or shared objects. It uses the runtime linker, `ld.so.1`, to generate the diagnostics, since it takes the object being inspected and prepares it as it would in a running process. By default, `ldd` triggers the loading of any lazy dependencies.

If `filename` is an executable file, `ldd` lists the path names of all shared objects that would be loaded when `filename` is loaded.

If `filename` is a shared object, `ldd` lists the path names of all shared objects that would be loaded when `filename` is loaded. `ldd` expects shared objects to have execute permission. If this is not the case, `ldd` will issue a warning before attempting to process the file.

`ldd` processes its input one file at a time. For each input file, `ldd` performs one of the following:
- Lists the object dependencies if they exist.
- Succeeds quietly if dependencies do not exist.
- Prints an error message if processing fails.

`ldd` can also check the compatibility of `filename` with the shared objects it uses. With each of the following options, `ldd` prints warnings for any unresolved symbol references that would occur if `filename` were executed.

- `-d` Check immediate references.
- `-r` Check both immediate and lazy references.

Only one of the above options can be specified during any single invocation of `ldd`. Immediate references are typically to data items used by the executable or shared object code, pointers to functions, and even calls to functions made from a position dependent shared object. Lazy references are typically calls to global functions made from a position independent shared object, or calls to external functions made from the executable. For more information on these types of reference, see “When Relocations Are Performed” in the Linker and Libraries Guide. Object loading can also be affected by relocation processing. See Lazy Loading under USAGE for more details.

- `-c` Disables any configuration file use. Configuration files may be employed to alter default search paths, provide a directory cache, and provide alternative object dependencies. See `crle(1)`.
- `-e envvar` Sets the environment variable `envvar`. This option is useful for experimenting with runtime linker environment variables that can adversely affect `ldd` itself.
- `-f` Forces `ldd` to check for an executable file that is not secure. When `ldd` is invoked by a super-user, by default it will not process any

---

**NAME**

`ldd` – list dynamic dependencies of executable files or shared objects

**SYNOPSIS**

```bash
```
executable that it finds not secure. An executable is not considered secure if the interpreter it specifies does not reside under /usr/lib or /etc/lib, or if the interpreter cannot be determined. See Security under USAGE.

-i       Displays the order of execution of initialization sections. The order discovered may be affected by use of the -d or -r options. See Initialization Order under USAGE.

-L       Enables lazy loading. This is the default mode of operation when the object under inspection is loaded as part of a process. In this case, any lazy dependencies, or filters, are only loaded into the process when reference is made to a symbol that is defined within the lazy object. The -d or -r options, together with the -L option, may be used to inspect the dependencies and their order of loading since it will occur in a running process.

-l       Forces the immediate processing of any filters so that all filtees, and their dependencies, are listed. The immediate processing of filters is now the default mode of operation for ldd. However, under this default any auxiliary filtees that cannot be found are silently ignored. Under the -l option, missing auxiliary filtees generate an error message.

-s       Displays the search path used to locate shared object dependencies.

-u       Displays any unused dependencies. When a symbol reference is bound to a dependency, that dependency is deemed used. This option is therefore only useful when symbol references are being checked. If the -r option is not in effect, the -u option is enabled.

Objects that are found to be unused when using the -r option should be removed as dependencies. They provide no references but result in unnecessary overhead when filename is loaded. Objects that are found to be unused when using the -d option are not immediately required when filename is loaded, and are therefore candidates for lazy loading (see Lazy Loading under USAGE).

-v       Displays all dependency relationships incurred when processing filename. This option also displays any dependency version requirements. See pvs(1).

Security  A super-user should use the -f option only if the executable to be examined is known to be trustworthy, because use of -f on an untrustworthy executable while super-user may compromise system security. If it is unknown whether or not the executable to be examined is trustworthy, a super-user should temporarily become a regular user and invoke ldd as that regular user.
Untrustworthy objects can be safely examined with `dump(1)` and with `adb(1)`, as long as the `:r` subcommand is not used. In addition, a non-super-user can use either the `:r` subcommand of `adb` or `truss(1)` to examine an untrustworthy executable without too much risk of compromise. To minimize risk when using `ldd`, `adb :r`, or `truss` on an untrustworthy executable, use the user id "nobody".

**Lazy Loading**

Objects that employ lazy loading techniques, either through directly specified lazy dependencies (see the `-z` lazyload option of `ld(1)`), or through filters (see the `-f` and `-F` options of `ld(1)`), may experience variations in `ldd` output due to the options they use. If an object expresses all its dependencies as lazy, the default operation of `ldd` will list all dependencies in the order in which they are recorded in that object:

```
example% ldd main
libelf.so.1 => /usr/lib/libelf.so.1
libnsl.so.1 => /usr/lib/libnsl.so.1
libc.so.1 => /usr/lib/libc.so.1
```

The lazy loading behavior that occurs when this object is used at runtime may be enabled using the `-L` option. In this mode, lazy dependencies are loaded when reference is made to a symbol that is defined within the lazy object. Therefore, combining the `-L` option with use of the `-d` and `-r` options will reveal the dependencies needed to satisfy the immediate and lazy references respectively:

```
example% ldd -L main
example% ldd -d main
lib.so.1 => /usr/lib/libc.so.1
libelf.so.1 => /usr/lib/libelf.so.1
```

```
example% ldd -r main
lib.so.1 => /usr/lib/libc.so.1
libelf.so.1 => /usr/lib/libelf.so.1
```

Notice that in this example, the order of the dependencies listed is not the same as displayed from `ldd` with no options, and even with the `-r` option, the lazy reference to dependencies may not occur in the same order as it will in a running program.

Observing lazy loading may also reveal objects that are not required to satisfy any references. These objects (in this example, `libnsl.so.1`) are candidates for removal from the link-line used to build the object being inspected.

Objects that do not explicitly define their required dependencies may observe variations in the initialization section order displayed by `ldd` due to the options they use. For example, a simple application may reveal:

```
example% ldd -i main
libA.so.1 => ./libA.so.1
libc.so.1 => /usr/lib/libc.so.1
libB.so.1 => ./libB.so.1
```

```
init object=./libB.so.1
init object=./libA.so.1
init object=/usr/lib/libc.so.1
```

whereas, when relocations are applied, the initialization section order is:
In this case, libB.so.1 makes reference to a function in /usr/lib/libc.so.1. However, it has no explicit dependency on this library. Only after a relocation is discovered is a dependency established, which in turn affects the initialization section sort order.

Typically, the initialization section sort order established when an application is executed is equivalent to ldd with the -d option. The optimum order can be obtained if all objects fully define their dependencies. Use of the ld(1) options -z defs and -z ignore when building dynamic objects is recommended.

Cyclic dependencies may result when one or more dynamic objects reference each other. Cyclic dependencies should be avoided, as a unique initialization sort order for these dependencies can not be established.

Users that prefer a more static analysis of object files may inspect dependencies using tools such as dump(1) and elfdump(1).

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWtoo</td>
</tr>
</tbody>
</table>

SEE ALSO

adb(1), crle(1), dump(1), elfdump(1), ld(1), ld.so.1(1), pvs(1), truss(1), dlopen(3DL), attributes(5)

Linker and Libraries Guide

DIAGNOSTICS

ldd prints the record of shared object path names to stdout. The optional list of symbol resolution problems is printed to stderr. If filename is not an executable file or a shared object, or if it cannot be opened for reading, a non-zero exit status is returned.

NOTES

ldd does not list shared objects explicitly attached using dlopen(3DL).
Using the `-d` or `-r` option with shared objects can give misleading results. `ldd` does a "worst case" analysis of the shared objects. However, in practice some or all of the symbols reported as unresolved can be resolved by the executable file referencing the shared object. The runtime linkers preloading mechanism (see `LD_PRELOAD`) may be employed to add dependencies to the object being inspected.

`ldd` uses the same algorithm as the runtime linker to locate shared objects.
Dynamic applications consist of one or more dynamic objects. They are typically a dynamic executable and its shared object dependencies. As part of the initialization and execution of a dynamic application, an interpreter is called to complete the binding of the application to its shared object dependencies. In Solaris, this interpreter is referred to as the runtime linker.

During the link-editing of a dynamic executable, a special .interp section, together with an associated program header, is created. This section contains a pathname specifying the program’s interpreter. The pathname to the interpreter can be specified when the executable is being constructed by the -I option to ld(1) the link-editor. The default name supplied by the link-editor is that of the runtime linker, /usr/lib/ld.so.1.

During the process of executing a dynamic executable the kernel maps the file and locates the required interpreter. See exec(2) and mmap(2). The kernel maps this interpreter and transfers control to it, passing sufficient information to allow the interpreter to continue binding the application and then run it.

In addition to initializing an application, the runtime linker provides services that allow the application to extend its address space by mapping additional shared objects and binding to symbols within them.

The runtime linker performs the following functions:

- It opens and processes any applicable configuration file. Configuration files may be employed to alter default search paths, provide a directory cache and provide alternative object dependencies. See crle(1). By default, the configuration file /var/ld/ld.config is used for 32-bit objects and /var/ld/64/ld.config for 64-bit objects. Alternative configuration files can be specified with the LD_CONFIG environment variable, or encoded within a dynamic executable using the -c option of ld(1).

- It analyzes the application’s dynamic information section (.dynamic) and determines which shared object dependencies are required.

- It locates and maps in these dependencies, and then it analyzes their dynamic information sections to determine if any additional shared object dependencies are required.

- Once all shared object dependencies are located and mapped, the runtime linker performs any necessary relocations to bind these shared objects in preparation for process execution.

- It calls any initialization functions provided by the shared object dependencies and, possibly, by the dynamic executable. By default, these are called in the reverse order of the topologically sorted dependencies. If cyclic dependencies exist, the initialization functions are called using the sorted order with the cycle removed.
ld.so.1(1)

ldd(1) can be used to display the initialization order of shared object dependencies.

- It passes control to the application.
- During the application’s execution, the runtime linker can be called upon to perform any delayed function binding.
- It calls any finalization functions on deletion of shared objects from the process. By default these are called in the order of the topologically sorted dependencies.
- The application can also call upon the runtime linker’s services to acquire additional shared objects by dlopen(3DL) and bind to symbols within these objects with dlsym(3DL)

Further details on each of the above topics may be found in the Linker and Libraries Guide.

The runtime linker uses a prescribed search path for locating the dynamic dependencies of an object. The default search paths are the runpath recorded in the object, followed by /usr/lib for 32-bit objects or /usr/lib/64 for 64-bit objects. This latter component can be modified using a configuration file created with crle(1). The runpath is specified when the dynamic object is constructed using the -R option to ld(1). LD_LIBRARY_PATH can be used to indicate directories to be searched before the default directories.

### ENVIRONMENT VARIABLES

- **LD_AUDIT**
  A colon-separated list of objects that will be loaded by the runtime linker. As each object is loaded, it will be examined for Link-Auditing interfaces; the routines that are present will be called as specified in the Link-Auditing interface described in the Linker and Libraries Guide. Also, see the -p and -P options of ld(1).

- **LD_BIND_NOW**
  The runtime linker’s default mode of performing lazy binding can be overridden by setting the environment variable LD_BIND_NOW to any non-null value. This setting causes the runtime linker to perform both data reference and function reference relocations during process initialization, before transferring control to the application. Also see the -z now option of ld(1).

- **LD_CONFIG**
  Provides an alternative configuration file. Configuration files may be employed to alter default search paths, provide a directory cache and provide alternate object dependencies. See crle(1).

- **LD_DEBUG**
  Provides a comma, or colon-separated list of tokens to cause the runtime linker to print debugging information to the standard error. The special token help indicates the full list of tokens available. The environment variable LD_DEBUG_OUTPUT may also be
supplied to specify a file to which the debugging information is sent. The filename will be suffixed with the process ID of the application generating the debugging information.

**LD_DEMANGLE** Any symbol name used as part of a diagnostic message is shown as it is defined within an ELF file. When **LD_DEMANGLE** is set to any non-null value, the runtime linker will attempt to decode (demangle) any C++ symbol name.

**LD_FLAGS** Provides an alternative means of supplying environment variable information. Any of the **LD_XXX** environment variables can specified as a *xxx* token. Multiple tokens can be supplied separated by commas. See EXAMPLES.

**LD_LIBRARY_PATH** The **LD_LIBRARY_PATH** environment variable, if set, is used to enhance the search path that the runtime linker uses to find dynamic dependencies. **LD_LIBRARY_PATH** specifies a colon-separated list of directories that are to be searched before the default directories. Also notice that **LD_LIBRARY_PATH** adds additional semantics to *ld(1)*.

**LD_LOADFLTR** Filters are a form of shared object. They allow an alternative shared object to be selected at runtime and provide the implementation for any symbols defined within the filter. See the `-f` and `-F` options of *ld(1)*. By default, the alternative shared object processing is deferred until symbol resolution occurs against the filter. When **LD_LOADFLTR** is set to any non-null value, the runtime linker will process filters immediately when they are loaded. Also see the `-z loadfltr` option of *ld(1)*.

**LD_NOAUDIT** Local auditing libraries can be defined within applications and shared objects. See the `-p` and `-P` options of *ld(1)*. When **LD_NOAUDIT** is set to any non-null value, the runtime linker will ignore any local auditing libraries.

**LD_NOAUXFLTR** Auxiliary filters are a form of shared object. They allow an alternative shared object to be selected at runtime which provides the implementation for any symbols defined within the filter. See the `-f` option of *ld(1)*. When **LD_NOAUXFLTR** is set to any non-null value, the runtime linker will disable this alternative shared object lookup.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LD_NOCONFIG</td>
<td>By default the runtime linker attempts to open and process a configuration file. When LD_NOCONFIG is set to any non-null value, the runtime linker will disable this configuration file processing.</td>
</tr>
<tr>
<td>LD_NODIRCONFIG</td>
<td>Provides a subset of LD_NOCONFIG in that any directory cache information provided in a configuration file is ignored.</td>
</tr>
<tr>
<td>LD_NODIRECT</td>
<td>Direct binding information instructs the runtime linker to search directly for a symbol in an associated object, rather than carry out the default symbol search model. See the -B direct option of ld(1). When LD_NODIRECT is set to any non-null value, the runtime linker will ignore any direct binding information.</td>
</tr>
<tr>
<td>LD_NOENVCONFIG</td>
<td>Provides a subset of LD_NOCONFIG in that any environment variables provided in a configuration file are ignored.</td>
</tr>
<tr>
<td>LD_NOLAZYLOAD</td>
<td>Dependencies labeled for lazy loading are not loaded into memory until explicit reference has been made to them. See the -z lazyload option of ld(1). When LD_NOLAZYLOAD is set to any non-null value, the runtime linker will ignore a dependencies lazy loading label and load it immediately.</td>
</tr>
<tr>
<td>LD_NOOBJALTER</td>
<td>Provides a subset of LD_NOCONFIG in that any alternative object dependencies provided in a configuration file are ignored.</td>
</tr>
<tr>
<td>LD_NOVERSION</td>
<td>By default the runtime linker verifies version dependencies for the primary executable and all of its dependencies. When LD_NOVERSION is set to any non-null value, the runtime linker will disable this version checking.</td>
</tr>
<tr>
<td>LD_ORIGIN</td>
<td>The immediate processing of $ORIGIN can be triggered by setting the environment variable LD_ORIGIN to any non-null value. Before Solaris 9, this option was useful for applications that invoked chdir(2) prior to locating dependencies that employed the $ORIGIN string token. The establishment of the current working directory by the runtime linker is now default and thus making this option redundant.</td>
</tr>
</tbody>
</table>
| LD_PRELOAD           | Provides a list of shared objects, separated by spaces, that are loaded after the program being executed but before any other shared objects that the program references. Symbol definitions provided by the preloaded objects interpose on references made by the
shared objects that the program references, but do not interpose on the program itself.

**LD_PROFILE**

Defines a shared object that will be profiled by the runtime linker. When profiling is enabled, a profiling buffer file is created and mapped. The name of the buffer file is the name of the shared object being profiled with a .profile extension. By default, this buffer is placed under /var/tmp. The environment variable LD_PROFILE_OUTPUT may also be supplied to indicate an alternative directory in which to place the profiling buffer.

This buffer contains profil(2) and call count information similar to the gmon.out information generated by programs that have been linked with the -xpg option of cc. Any applications that use the named shared object and run while this environment variable is set will accumulate data in the profile buffer. See also NOTES. The profile buffer information may be examined using gprof(1).

Notice that this profiling technique is an alternative to any that may be provided by the compilation system. The shared object being profiled does not have to be instrumented in any way, and LD_PROFILE should not be combined with a profile-instrumented application. See the Linker and Libraries Guide for more information on profiling shared objects.

**LD_SIGNAL**

Provides a numeric signal number that the runtime linker will use to kill the process in the event of a fatal runtime error (see kill(2)). By default, SIGKILL is used. For example, providing the alternative signal number 6 (SIGABRT), can provide for the creation of a core file to aid debugging.

Each environment variable can be specified with a _32 or _64 suffix. This makes the environment variable specific, respectively, to 32-bit or 64-bit processes and overrides any non-suffixed version of the environment variable that may be in effect.

Notice that environment variable names beginning with the characters ‘LD_’ are reserved for possible future enhancements to ld(1) and ld.so.1.

**SECURITY**

Secure processes have some restrictions applied to the evaluation of their dependencies and runpaths to prevent malicious dependency substitution or symbol interposition.
The runtime linker categorizes a process as secure if the user is not a super-user, and either the real user and effective user identifiers are not equal, or the real group and effective group identifiers are not equal. See getuid(2), geteuid(2), getgid(2), and getegid(2).

The default trusted directory known to the runtime linker is /usr/lib/secure for 32-bit objects or /usr/lib/secure/64 for 64-bit objects. The utility crle(1) may be used to specify additional trusted directories applicable for secure applications. Administrators who use this technique should ensure that the target directories are suitably protected from malicious intrusion.

If an LD_LIBRARY_PATH environment variable is in effect for a secure process, then only the trusted directories specified by this variable will be used to augment the runtime linker’s search rules.

In a secure process, any runpath specifications provided by the application or any of its dependencies will be used, provided they are full pathnames, that is, the pathname starts with a ‘/’.

In a secure process, the expansion of the $ORIGIN string is allowed only if it expands to a trusted directory.

In a secure process, LD_SIGNAL is ignored.

Additional objects may be loaded with a secure process using the LD_PRELOAD, or LD_AUDIT environment variables. These objects must be specified as full pathnames or simple file names. Full pathnames are restricted to known trusted directories. Simple file names, in which no ‘/’ appears in the name, are located subject to the search path restrictions previously described, and thus will only resolve to known trusted directories.

In a secure process, any dependencies that consist of simple filenames will be processed using the pathname restrictions previously described. Dependencies that are expressed as full or relative pathnames will be used as is. Therefore, the developer of a secure process should ensure that the target directory referenced as a full or relative pathname dependency is suitably protected from malicious intrusion.

When creating a secure process, it is recommended that relative pathnames not be used to express dependencies or to construct dlopen(3DL) pathnames. This restriction should be applied to the application and to all dependencies.

**EXAMPLE 1** Using LD_FLAGS to group environment variable information

The following use of LD_FLAGS,

```bash
example$ LD_FLAGS_32=bind_now,library_path=/lib/one:/lib/two
example$ LD_FLAGS_64=library_path=/lib/one/64,preload=foo.so
```

ld.so.1(1)
EXAMPLE 1 Using LD_FLAGS to group environment variable information

(Continued)

is equivalent to setting the individual environment variables LD_BIND_NOW and LD_LIBRARY_PATH for 32-bit applications, and LD_LIBRARY_PATH and LD_PRELOAD for 64-bit applications.

FILES

/usr/lib/ld.so.1 Default runtime linker.
/etc/lib/ld.so.1 Alternate runtime linker.
/usr/lib/libc.so.1 Alternate interpreter for SVID ABI compatibility.
/usr/lib/ld.so AOUT (BCP) runtime linker.
/usr/lib/0@0.so.1 Null character pointer compatibility library. See NOTES.
/usr/lib/secure LD_PRELOAD location for secure applications.
/usr/lib/secure/64 LD_PRELOAD location for secure 64-bit applications.
/usr/lib/64/ld.so.1 Default runtime linker for 64-bit applications.
/usr/lib/64/0@0.so.1 Null character pointer compatibility library for the 64-bit applications.
/var/ld/ld.config Default configuration file for 32-bit applications.
/var/ld/64/ld.config Default configuration file for 64-bit applications.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO
crl(1), gprof(1), ld(1), ldd(1), exec(2), getegid(2), geteuid(2), getuid(2), kill(2), mmap(2), profil(2), dladdr(3DL), dlclose(3DL), dlEither(3DL), dlerror(3DL), dlstrcmp(3DL), dlerror(3DL), dlerror(3DL), proc(4), attributes(5)

Linker and Libraries Guide

NOTES Care should be exercised when using LD_PROFILE in combination with other process monitoring techniques (for example, users of proc(4)) as deadlock conditions can result that leave the profile buffer locked. A locked buffer will block any processes that
try to record profiling information. To reduce this likelihood, the runtime linker’s profile implementation determines if the process is being monitored at startup, and if so, silently disables profiling of the process. However, this mechanism can not catch monitoring processes that attach to the process during its execution.

The user compatibility library /usr/lib/0@0.so.1 provides a mechanism that establishes a value of 0 at location 0. Some applications exist that erroneously assume a null character pointer should be treated the same as a pointer to a null string. A segmentation violation will occur in these applications when a null character pointer is accessed. If this library is added to such an application at runtime using LD_PRELOAD, it provides an environment that is sympathetic to this errant behavior. However, the user compatibility library is intended neither to enable the generation of such applications, nor to endorse this particular programming practice.
NAME
let – shell built-in function to evaluate one or more arithmetic expressions

ksh
let arg...

ksh
Each arg is a separate "arithmetic expression" to be evaluated.

EXIT STATUS
The following exit values are returned:

0 The value of the last expression is non-zero.
1 The value of the last expression is zero.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO
ksh(1), set(1), typeset(1), attributes(5)
lex(1)

NAME
lex – generate programs for lexical tasks

SYNOPSIS
lex [-cntv] [-e | -w] [-V -Q [y | n]] [file...]

DESCRIPTION
The lex utility generates C programs to be used in lexical processing of character
input, and that can be used as an interface to yacc. The C programs are generated
from lex source code and conform to the ISO C standard. Usually, the lex utility
writes the program it generates to the file lex.yy.c; the state of this file is
unspecified if lex exits with a non-zero exit status. See EXTENDED DESCRIPTION for
a complete description of the lex input language.

OPTIONS
The following options are supported:

-c
  Indicate C-language action (default option).
-e
  Generate a program that can handle EUC characters (cannot be
  used with the -w option).
  yytext[ ] is of type unsigned
  char[ ].
-n
  Suppress the summary of statistics usually written with the -v
  option. If no table sizes are specified in the lex source code and
  the -v option is not specified, then -n is implied.
-t
  Write the resulting program to standard output instead of
  lex.yy.c.
-v
  Write a summary of lex statistics to the standard error. (See the
  discussion of lex table sizes under the heading Definitions in
  lex.) If table sizes are specified in the lex source code, and if the
  -n option is not specified, the -v option may be enabled.
-w
  Generate a program that can handle EUC characters (cannot be
  used with the -e option).
  Unlike the -e option, yytext[ ] is of
  type wchar_t[ ].
-V
  Print out version information on standard error.
-Q[y|n]
  Print out version information to output file lex.yy.c by using
  -Qy. The -Qn option does not print out version information and is
  the default.

OPERANDS
The following operand is supported:

file
  A pathname of an input file. If more than one such file is specified, all
  files will be concatenated to produce a single lex program. If no file
  operands are specified, or if a file operand is −, the standard input will be
  used.

Stdout
  If the -t option is specified, the text file of C source code output of lex will be written
  to standard output.

Stderr
  If the -t option is specified informational, error and warning messages concerning the
  contents of lex source code input will be written to the standard error.

686 man pages section 1: User Commands • Last Revised 22 Aug 1997
If the -t option is not specified:

1. Informational error and warning messages concerning the contents of lex source code input will be written to either the standard output or standard error.

2. If the -v option is specified and the -n option is not specified, lex statistics will also be written to standard error. These statistics may also be generated if table sizes are specified with a % operator in the Definitions in lex section (see EXTENDED DESCRIPTION), as long as the -n option is not specified.

Output Files

A text file containing C source code will be written to lex.yy.c, or to the standard output if the -t option is present.

EXTENDED DESCRIPTION

Each input file contains lex source code, which is a table of regular expressions with corresponding actions in the form of C program fragments.

When lex.yy.c is compiled and linked with the lex library (using the -l l operand with c89 or cc), the resulting program reads character input from the standard input and partitions it into strings that match the given expressions.

When an expression is matched, these actions will occur:

- The input string that was matched is left in yytext as a null-terminated string; yytext is either an external character array or a pointer to a character string. As explained in Definitions in lex, the type can be explicitly selected using the %array or %pointer declarations, but the default is %array.
- The external int yyleng is set to the length of the matching string.
- The expression’s corresponding program fragment, or action, is executed.

During pattern matching, lex searches the set of patterns for the single longest possible match. Among rules that match the same number of characters, the rule given first will be chosen.

The general format of lex source is:

Definitions
%%
Rules
%%
User Subroutines

The first %% is required to mark the beginning of the rules (regular expressions and actions); the second %% is required only if user subroutines follow.

Any line in the Definitions in lex section beginning with a blank character will be assumed to be a C program fragment and will be copied to the external definition area of the lex.yy.c file. Similarly, anything in the Definitions in lex section included between delimiter lines containing only %{ and %} will also be copied unchanged to the external definition area of the lex.yy.c file.
Any such input (beginning with a blank character or within \{ and \} delimiter lines) appearing at the beginning of the Rules section before any rules are specified will be written to \texttt{lex.yy.c} after the declarations of variables for the \texttt{yylex} function and before the first line of code in \texttt{yylex}. Thus, user variables local to \texttt{yylex} can be declared here, as well as application code to execute upon entry to \texttt{yylex}.

The action taken by \texttt{lex} when encountering any input beginning with a blank character or within \{ and \} delimiter lines appearing in the Rules section but coming after one or more rules is undefined. The presence of such input may result in an erroneous definition of the \texttt{yylex} function.

### Definitions in lex

Definitions in \texttt{lex} appear before the first \texttt{%%} delimiter. Any line in this section not contained between \{ and \} lines and not beginning with a blank character is assumed to define a \texttt{lex} substitution string. The format of these lines is:

\texttt{name substitute}

If a \texttt{name} does not meet the requirements for identifiers in the ISO C standard, the result is undefined. The string \texttt{substitute} will replace the string \{ \texttt{name} \} when it is used in a rule. The \texttt{name} string is recognized in this context only when the braces are provided and when it does not appear within a bracket expression or within double-quotes.

In the Definitions in \texttt{lex} section, any line beginning with a \% (percent sign) character and followed by an alphanumeric word beginning with either s or S defines a set of start conditions. Any line beginning with a \% followed by a word beginning with either x or X defines a set of exclusive start conditions. When the generated scanner is in a \%s state, patterns with no state specified will be also active; in a \%x state, such patterns will not be active. The rest of the line, after the first word, is considered to be one or more blank-character-separated names of start conditions. Start condition names are constructed in the same way as definition names. Start conditions can be used to restrict the matching of regular expressions to one or more states as described in Regular expressions in \texttt{lex}.

Implementations accept either of the following two mutually exclusive declarations in the Definitions in \texttt{lex} section:

\begin{itemize}
  \item \texttt{array} \quad Declare the type of \texttt{yytext} to be a null-terminated character array.
  \item \texttt{pointer} \quad Declare the type of \texttt{yytext} to be a pointer to a null-terminated character string.
\end{itemize}

Note: When using the \texttt{pointer} option, you may not also use the \texttt{yyless} function to alter \texttt{yytext}.

\texttt{array} is the default. If \texttt{array} is specified (or neither \texttt{array} nor \texttt{pointer} is specified), then the correct way to make an external reference to \texttt{yytext} is with a declaration of the form:

\begin{verbatim}
lex(1)
\end{verbatim}
extern char yytext[]

If %pointer is specified, then the correct external reference is of the form:

extern char *yytext;

lex will accept declarations in the Definitions in lex section for setting certain internal table sizes. The declarations are shown in the following table.

Table: Size Declaration in lex

<table>
<thead>
<tr>
<th>Declaration</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>%pn</td>
<td>Number of positions</td>
<td>2500</td>
</tr>
<tr>
<td>%nn</td>
<td>Number of states</td>
<td>500</td>
</tr>
<tr>
<td>%an</td>
<td>Number of transitions</td>
<td>2000</td>
</tr>
<tr>
<td>%en</td>
<td>Number of parse tree nodes</td>
<td>1000</td>
</tr>
<tr>
<td>%kn</td>
<td>Number of packed character classes</td>
<td>10000</td>
</tr>
<tr>
<td>%on</td>
<td>Size of the output array</td>
<td>3000</td>
</tr>
</tbody>
</table>

Programs generated by lex need either the -e or -w option to handle input that contains EUC characters from supplementary codesets. If neither of these options is specified, yytext is of the type char[], and the generated program can handle only ASCII characters.

When the -e option is used, yytext is of the type unsigned char[] and yyleng gives the total number of bytes in the matched string. With this option, the macros input(), unput(c), and output(c) should do a byte-based I/O in the same way as with the regular ASCII lex. Two more variables are available with the -e option, yywtext and yywleng, which behave the same as yytext and yyleng would under the -w option.

When the -w option is used, yytext is of the type wchar_t[] and yyleng gives the total number of characters in the matched string. If you supply your own input(), unput(c), or output(c) macros with this option, they must return or accept EUC characters in the form of wide character (wchar_t). This allows a different interface between your program and the lex internals, to expedite some programs.

Rules in lex

The Rules in lex source files are a table in which the left column contains regular expressions and the right column contains actions (C program fragments) to be executed when the expressions are recognized.

ERE action
ERE action
The extended regular expression (ERE) portion of a row will be separated from action by one or more blank characters. A regular expression containing blank characters is recognized under one of the following conditions:

- The entire expression appears within double-quotes.
- The blank characters appear within double-quotes or square brackets.
- Each blank character is preceded by a backslash character.

Anything in the user subroutines section will be copied to lex.yy.c following yylex.

The lex utility supports the set of Extended Regular Expressions (EREs) described on regex(5) with the following additions and exceptions to the syntax:

- Any string enclosed in double-quotes will represent the characters within the double-quotes as themselves, except that backslash escapes (which appear in the following table) are recognized. Any backslash-escape sequence is terminated by the closing quote. For example, " \ 01" represents a single string: the octal value 1 followed by the character 1.

\( <\text{state}> \) \( r \)

\( <\text{state1, state2, ...}> \) \( r \)

The regular expression \( r \) will be matched only when the program is in one of the start conditions indicated by \( \text{state, state1,} \) and so forth; for more information see Actions in lex (As an exception to the typographical conventions of the rest of this document, in this case \( <\text{state}> \) does not represent a metavariable, but the literal angle-bracket characters surrounding a symbol.) The start condition is recognized as such only at the beginning of a regular expression.

\( r/x \)

The regular expression \( r \) will be matched only if it is followed by an occurrence of regular expression \( x \). The token returned in yytext will only match \( r \). If the trailing portion of \( r \) matches the beginning of \( x \), the result is unspecified. The \( r \) expression cannot include further trailing context or the $ (match-end-of-line) operator; \( x \) cannot include the ^ (match-beginning-of-line) operator, nor trailing context, nor the $ operator. That is, only one occurrence of trailing context is allowed in a lex regular expression, and the ^ operator only can be used at the beginning of such an expression. A further restriction is that the trailing-context operator / (slash) cannot be grouped within parentheses.

\{name\}

When \( \text{name} \) is one of the substitution symbols from the Definitions section, the string, including the enclosing braces, will be replaced by the substitute value. The substitute value will be treated in the extended regular expression as if it were enclosed in parentheses. No substitution will occur if \( \{\text{name}\} \) occurs within a bracket expression or within double-quotes.
Within an ERE, a backslash character (\, \a, \b, \f, \n, \r, \t, \v) is considered to begin an escape sequence. In addition, the escape sequences in the following table will be recognized.

A literal newline character cannot occur within an ERE; the escape sequence \n can be used to represent a newline character. A newline character cannot be matched by a period operator.

**Escape Sequences in lex**

<table>
<thead>
<tr>
<th>Escape Sequence</th>
<th>Description</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>\digits</td>
<td>A backslash character followed by the longest sequence of one, two or three octal-digit characters (01234567). If all of the digits are 0, (that is, representation of the NUL character), the behavior is undefined.</td>
<td>The character whose encoding is represented by the one-, two- or three-digit octal integer. Multi-byte characters require multiple, concatenated escape sequences of this type, including the leading \ for each byte.</td>
</tr>
<tr>
<td>\xdigits</td>
<td>A backslash character followed by the longest sequence of hexadecimal-digit characters (01234567abcdefABCDEF). If all of the digits are 0, (that is, representation of the NUL character), the behavior is undefined.</td>
<td>The character whose encoding is represented by the hexadecimal integer.</td>
</tr>
<tr>
<td>\c</td>
<td>A backslash character followed by any character not described in this table. (, \a, \b, \f, \n, \r, \t, \v)</td>
<td>The character c, unchanged.</td>
</tr>
</tbody>
</table>

The order of precedence given to extended regular expressions for lex is as shown in the following table, from high to low.

**Note:** The escaped characters entry is not meant to imply that these are operators, but they are included in the table to show their relationships to the true operators. The start condition, trailing context and anchoring notations have been omitted from the table because of the placement restrictions described in this section; they can only appear at the beginning or ending of an ERE.
**ERE Precedence in lex**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>collation-related bracket symbols</td>
<td>[= =] [[: :] [. .]</td>
</tr>
<tr>
<td>escaped characters</td>
<td>&lt;special character&gt;</td>
</tr>
<tr>
<td>bracket expression</td>
<td>[ ]</td>
</tr>
<tr>
<td>quoting</td>
<td>&quot;...&quot;</td>
</tr>
<tr>
<td>grouping</td>
<td>{}</td>
</tr>
<tr>
<td>definition</td>
<td>{name}</td>
</tr>
<tr>
<td>single-character RE duplication</td>
<td>* + ?</td>
</tr>
<tr>
<td>concatenation</td>
<td></td>
</tr>
<tr>
<td>interval expression</td>
<td>{m,n}</td>
</tr>
<tr>
<td>alternation</td>
<td></td>
</tr>
</tbody>
</table>

The ERE anchoring operators ( ^ and $) do not appear in the table. With lex regular expressions, these operators are restricted in their use: the ^ operator can only be used at the beginning of an entire regular expression, and the $ operator only at the end. The operators apply to the entire regular expression. Thus, for example, the pattern (^abc) | (def$) is undefined; it can instead be written as two separate rules, one with the regular expression ^abc and one with def$, which share a common action via the special | action (see below). If the pattern were written ^abc | def$, it would match either of abc or def on a line by itself.

Unlike the general ERE rules, embedded anchoring is not allowed by most historical lex implementations. An example of embedded anchoring would be for patterns such as (^)foo($) to match foo when it exists as a complete word. This functionality can be obtained using existing lex features:

```
^foo/\[ \ n\]|
" foo"/\[ \ n\]  /* found foo as a separate word */
```

Note also that $ is a form of trailing context (it is equivalent to /\ n and as such cannot be used with regular expressions containing another instance of the operator (see the preceding discussion of trailing context).

The additional regular expressions trailing-context operator / (slash) can be used as an ordinary character if presented within double-quotes, " / "; preceded by a backslash, \ /; or within a bracket expression, [ / ]. The start-condition < and > operators are special only in a start condition at the beginning of a regular expression; elsewhere in the regular expression they are treated as ordinary characters.

The following examples clarify the differences between lex regular expressions and regular expressions appearing elsewhere in this document. For regular expressions of the form r/x, the string matching r is always returned; confusion may arise when the
beginning of \( x \) matches the trailing portion of \( r \). For example, given the regular expression \( a^*b/cc \) and the input \( aaaabcc \), \( yytext \) would contain the string \( aaaab \) on this match. But given the regular expression \( x^*/xy \) and the input \( xxxxy \), the token \( xxx \), not \( xx \), is returned by some implementations because \( xxx \) matches \( x^* \).

In the rule \( ab^*/bc \), the \( b^* \) at the end of \( r \) will extend \( r \)'s match into the beginning of the trailing context, so the result is unspecified. If this rule were \( ab/bc \), however, the rule matches the text \( ab \) when it is followed by the text \( bc \). In this latter case, the matching of \( r \) cannot extend into the beginning of \( x \), so the result is specified.

**Actions in `lex`**

The action to be taken when an ERE is matched can be a C program fragment or the special actions described below; the program fragment can contain one or more C statements, and can also include special actions. The empty C statement \( ; \) is a valid action; any string in the `lex . yy . c` input that matches the pattern portion of such a rule is effectively ignored or skipped. However, the absence of an action is not valid, and the action `lex` takes in such a condition is undefined.

The specification for an action, including C statements and special actions, can extend across several lines if enclosed in braces:

```c
ERE <one or more blanks> { program statement
  program statement }```

The default action when a string in the input to a `lex . yy . c` program is not matched by any expression is to copy the string to the output. Because the default behavior of a program generated by `lex` is to read the input and copy it to the output, a minimal `lex` source program that has just `%%` generates a C program that simply copies the input to the output unchanged.

Four special actions are available:

`| ECHO; REJECT; BEGIN`

`|`

The action `|` means that the action for the next rule is the action for this rule. Unlike the other three actions, `|` cannot be enclosed in braces or be semicolon-terminated; it must be specified alone, with no other actions.

`ECHO;`

Write the contents of the string `yytext` on the output.

`REJECT;`

Usually only a single expression is matched by a given string in the input. `REJECT` means "continue to the next expression that matches the current input," and causes whatever rule was the second choice after the current rule to be executed for the same input. Thus, multiple rules can be matched and executed for one input string or overlapping input strings. For example, given the regular expressions `xyz` and `xy` and the input `xyz`, usually only the regular expression `xyz` would match. The next attempted match would start after `z`. If the last action in the `xyz` rule is
**REJECT**, both this rule and the *xy* rule would be executed. The
*REJECT* action may be implemented in such a fashion that flow of
control does not continue after it, as if it were equivalent to a *goto* to another part of *yylex*. The use of *REJECT* may result in
somewhat larger and slower scanners.

**BEGIN**

The action:

```c
BEGIN newstate;
```

switches the state (start condition) to *newstate*. If the string *newstate*
has not been declared previously as a start condition in the
*Definitions in lex* section, the results are unspecified. The
initial state is indicated by the digit 0 or the token *INITIAL*.

The functions or macros described below are accessible to user code included in the
*lex* input. It is unspecified whether they appear in the C code output of *lex*, or are
accessible only through the `-ll` operand to *c89* or *cc* (the *lex* library).

- **int yylex(void)**: Performs lexical analysis on the input; this is the
  primary function generated by the *lex* utility. The
  function returns zero when the end of input is reached;
  otherwise it returns non-zero values (tokens)
  determined by the actions that are selected.

- **int yymore(void)**: When called, indicates that when the next input string
  is recognized, it is to be appended to the current value
  of *yytext* rather than replacing it; the value in *yyleng* is
  adjusted accordingly.

- **int yyless(int n)**: Retains *n* initial characters in *yytext*, NUL-terminated,
  and treats the remaining characters as if they had not
  been read; the value in *yyleng* is adjusted accordingly.

- **int input(void)**: Returns the next character from the input, or zero on
  end-of-file. It obtains input from the stream pointer
  *yyin*, although possibly via an intermediate buffer.
  Thus, once scanning has begun, the effect of altering
  the value of *yyin* is undefined. The character read is
  removed from the input stream of the scanner without any
  processing by the scanner.

- **int unput(int c)**: Returns the character *c* to the input; *yytext* and *yyleng*
  are undefined until the next expression is matched. The
  result of using *unput* for more characters than have
  been input is unspecified.

The following functions appear only in the *lex* library accessible through the `-ll` operand; they can therefore be redefined by a portable application:
int yywrap(void)
    Called by yylex at end-of-file; the default yywrap always will return 1. If the
    application requires yylex to continue processing with another source of input,
    then the application can include a function yywrap, which associates another file
    with the external variable FILE *yyin and will return a value of zero.

int main(int argc, char *argv[])
    Calls yylex to perform lexical analysis, then exits. The user code can contain main
    to perform application-specific operations, calling yylex as applicable.

The reason for breaking these functions into two lists is that only those functions in
libl.a can be reliably redefined by a portable application.

Except for input, unput and main, all external and static names generated by lex
begin with the prefix yy or YY.

**USAGE**

Portable applications are warned that in the Rules in lex section, an ERE without an
action is not acceptable, but need not be detected as erroneous by lex. This may result
in compilation or run-time errors.

The purpose of input is to take characters off the input stream and discard them as
far as the lexical analysis is concerned. A common use is to discard the body of a
comment once the beginning of a comment is recognized.

The lex utility is not fully internationalized in its treatment of regular expressions in
the lex source code or generated lexical analyzer. It would seem desirable to have the
lexical analyzer interpret the regular expressions given in the lex source according to
the environment specified when the lexical analyzer is executed, but this is not
possible with the current lex technology. Furthermore, the very nature of the lexical
analyzers produced by lex must be closely tied to the lexical requirements of the
input language being described, which will frequently be locale-specific anyway. (For
example, writing an analyzer that is used for French text will not automatically be
useful for processing other languages.)

**EXAMPLES**

**EXAMPLE 1 Using lex**

The following is an example of a lex program that implements a rudimentary scanner
for a Pascal-like syntax:

```c
{%
/* need this for the call to atof() below */
#include <math.h>
/* need this for printf(), fopen() and stdin below */
#include <stdio.h>
%

DIGIT [0-9]
ID [a-z][a-z0-9]*
%%

{DIGIT}+ {
    printf("An integer: %s (%d)\n", yytext, atoi(yytext));
}
```
EXAMPLE 1 Using lex  (Continued)

    }

{DIGIT}+"."{DIGIT}*
    
    if|then|begin|end|procedure|function
        
        printf("A float: %s (%g)
    , yytext,
        atof(yytext));
    

{ID} printf("An identifier: %s
    , yytext);
"*"*"*"*/" printf("An operator: %s
    , yytext);
"{[^}]*}*/" /* eat up one-line comments */
[ \t\n]+ /* eat up white space */

    . printf("Unrecognized character: %s
    , yytext);

%%

int main(int argc, char *argv[ ])

    ++argv, --argc; /* skip over program name */
    if (argc > 0)
        yyin = fopen(argv[0], "r");
    else
        yyin = stdin;
    yylex();

ENVIRONMENT VARIABLES
See environ(5) for descriptions of the following environment variables that affect the execution of lex: LC_COLLATE, LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS
The following exit values are returned:

0    Successful completion.
>0    An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWbtool</td>
</tr>
</tbody>
</table>

SEE ALSO  yacc(1), attributes(5), environ(5), regex(5)
If routines such as yyback(), yywrap(), and yylock() in .l (ell) files are to be external C functions, the command line to compile a C++ program must define the __EXTERN_C__ macro. For example:

```bash
example$ cc -D__EXTERN_C__ ... file
```
limit(1)

NAME  limit, ulimit, unlimit – set or get limitations on the system resources available to the current shell and its descendents

SYNOPSIS  

/usr/bin/ulimit [-f] [blocks]

sh  ulimit [- [HS] [a | cd
dnfstv]]

sh  ulimit [- [HS] [c | d | f | n | s | t | v]] limit

csh  limit [-h] [resource [limit]]

csh  unlimit [-h] [resource]

ksh  ulimit [-HSacdfnstv] [limit]

The ulimit utility sets or reports the file-size writing limit imposed on files written by the shell and its child processes (files of any size may be read). Only a process with appropriate privileges can increase the limit.

The Bourne shell built-in function, ulimit, prints or sets hard or soft resource limits. These limits are described in getrlimit(2).

If limit is not present, ulimit prints the specified limits. Any number of limits may be printed at one time. The -a option prints all limits.

If limit is present, ulimit sets the specified limit to limit. The string unlimited requests the largest valid limit. Limits may be set for only one resource at a time. Any user may set a soft limit to any value below the hard limit. Any user may lower a hard limit. Only a super-user may raise a hard limit; see su(1M).

The -H option specifies a hard limit. The -S option specifies a soft limit. If neither option is specified, ulimit will set both limits and print the soft limit.

The following options specify the resource whose limits are to be printed or set. If no option is specified, the file size limit is printed or set.

- c maximum core file size (in 512-byte blocks)
- d maximum size of data segment or heap (in kbytes)
- f maximum file size (in 512-byte blocks)
- n maximum file descriptor plus 1
- s maximum size of stack segment (in kbytes)
- t maximum CPU time (in seconds)
- v maximum size of virtual memory (in kbytes)
The C-shell built-in function, `limit`, limits the consumption by the current process or any process it spawns, each not to exceed `limit` on the specified `resource`. If `limit` is omitted, print the current limit; if `resource` is omitted, display all limits. (Run the `sysdef(1M)` command to obtain the maximum possible limits for your system. The values reported are in hexadecimal, but can be translated into decimal numbers using the `bc(1)` command).

- `-h` Use hard limits instead of the current limits. Hard limits impose a ceiling on the values of the current limits. Only the privileged user may raise the hard limits.

`resource` is one of:

- `cputime` Maximum CPU seconds per process.
- `filesize` Largest single file allowed; limited to the size of the filesystem (see `df(1M)`).
- `datasize` The maximum size of a process’s heap in kilobytes.
- `stacksize` Maximum stack size for the process (see `swap(1M)`).
- `coredumpsize` Maximum size of a core dump (file). This is limited to the size of the filesystem.
- `descriptors` Maximum number of file descriptors (run `sysdef( )`).
- `memorysize` Maximum size of virtual memory.

`limit` is a number, with an optional scaling factor, as follows:

- `-h` Remove corresponding hard limits. Only the privileged user may do this.

The Korn shell built-in function, `ulimit`, sets or displays a resource limit. The available resources limits are listed below. Many systems do not contain one or more of these limits. The limit for a specified resource is set when `limit` is specified. The value of `limit` can be a number in the unit specified below with each resource, or the value `unlimited`. The `-H` and `-S` flags specify whether the hard limit or the soft limit for the given resource is set. A hard limit cannot be increased once it is set. A soft limit can be increased up to the value of the hard limit. If neither the `-H` or `-S` options is
specified, the limit applies to both. The current resource limit is printed when limit is omitted. In this case, the soft limit is printed unless -H is specified. When more than one resource is specified, then the limit name and unit is printed before the value.

- a Lists all of the current resource limits.
- c The number of 512-byte blocks on the size of core dumps.
- d The number of K-bytes on the size of the data area.
- f The number of 512-byte blocks on files written by child processes (files of any size may be read).
- n The number of file descriptors plus 1.
- s The number of K-bytes on the size of the stack area.
- t The number of seconds (CPU time) to be used by each process.
- v The number of K-bytes for virtual memory.

If no option is given, -f is assumed.

**OPTIONS**
The following option is supported by ulimit:

- f Set (or report, if no blocks operand is present), the file size limit in blocks.
The -f option is also the default case.

**OPERANDS**
The following operand is supported by ulimit:

blocks The number of 512-byte blocks to use as the new file size limit.

**EXAMPLE 1** Limiting the stack size

To limit the stack size to 512 kilobytes:

```
% ulimit -s 512
% ulimit -a
% time(seconds) unlimited
  file(blocks) 100
  data(kbytes) 523256
  stack(kbytes) 512
  coredump(blocks) 200
 nofiles(descriptors) 64
  memory(kbytes) unlimited
```

**EXAMPLE 2** Limiting the number of file descriptors

To limit the number of file descriptors to 12:

```
$ ulimit -n 12
$ ulimit -a
  time(seconds) unlimited
  file(blocks) 41943
  data(kbytes) 523256
  stack(kbytes) 8192
```

700 man pages section 1: User Commands • Last Revised 26 Jun 1998
EXAMPLE 2 Limiting the number of file descriptors (Continued)

coredump (blocks) 200
nofiles (descriptors) 12
vmemory (kbytes) unlimited

csh
EXAMPLE 3 Limiting the core dump file size

To limit the size of a core dump file size to 0 kilobytes:

% limit coredumpsize 0
% limit
cputime unlimited
filesize unlimited
datasize 523256 kbytes
stacksize 8192 kbytes
coredumpsize 0 kbytes
descriptors 64
memorysize unlimited

EXAMPLE 4 Removing the limitation for core file size

To remove the above limitation for the core file size:

% unlimit coredumpsize
% limit
cputime unlimited
filesize unlimited
datasize 523256 kbytes
stacksize 8192 kbytes
coredumpsize unlimited
descriptors 64
memorysize unlimited

ENVIRONMENT VARIABLES
See environ(5) for descriptions of the following environment variables that affect the execution of ulimit: LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS
The following exit values are returned by ulimit:

0  Successful completion.
>0  A request for a higher limit was rejected or an error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO
bc(1), csh(1), ksh(1), sh(1), df(1M), su(1M), swap(1M), sysdef(1M), getrlimit(2), attributes(5), environ(5)
NAME
line – read one line

SYNOPSIS
line

DESCRIPTION
The line utility copies one line (up to and including a new-line) from the standard input and writes it on the standard output. It returns an exit status of 1 on EOF and always prints at least a new-line. It is often used within shell files to read from the user’s terminal.

EXIT STATUS
Exit status is:

0    Successful completion
>0   End-of-file on input.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO
sh(1), read(2), attributes(5)
NAME      lint – C program verifier
SYNOPSIS  /usr/ucb/lint [options]
DESCRIPTION /usr/ucb/lint is the interface to the BSD Compatibility Package C program
     verifier. It is a script that looks for the link /usr/ccs/bin/ucblint to the C
     program verifier. /usr/ccs/bin/ucblint is available only with the SPROcc
     package, whose default location is /opt/SUNWspro. /usr/ucb/lint is identical to
     /usr/ccs/bin/ucblint, except that BSD headers are used and BSD libraries are
     linked before base libraries. The /opt/SUNWspro/man/man1/lint.1 man page is
     available only with the SPROcc package.
OPTIONS    /usr/ucb/lint accepts the same options as /usr/ccs/bin/ucblint, with the
     following exceptions:

     -Idir    Search dir for included files whose names do not begin with a
              slash ( / ) prior to searching the usual directories. The directories
              for multiple -I options are searched in the order specified. The
              preprocessor first searches for #include files in the directory
              containing sourcefile, and then in directories named with -I
              options (if any), then /usr/ucbininclude, and finally, in
              /usr/include.

     -Ldir    Add dir to the list of directories searched for libraries by
              /usr/ccs/bin/ucblint. This option is passed to
              /usr/ccs/bin/ld. Directories specified with this option are
              searched before /usr/ucblib and /usr/lib.

     -Y P, dir Change the default directory used for finding libraries.

EXIT STATUS The following exit values are returned:
     0        Successful completion.
     >0       An error occurred.
FILES      /usr/lint/bin/ld     link editor
            /usr/lib/libc       C library
            /usr/ucbininclude  BSD Compatibility directory for header files
            /usr/ucblib        BSD Compatibility directory for libraries
            /usr/ucblib/libucb  BSD Compatibility C library
            /usr/lib/libsocket library containing socket routines
            /usr/lib/libnsl    library containing network functions
            /usr/lib/libelf    library containing routines to process ELF object files
            /usr/lib/libaio    library containing synchronous I/O routines

ATTRIBUTES See attributes(5) for descriptions of the following attributes:
lint(1B)

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

SEE ALSO 1d(l), a.out(4), attributes(5)
### list_devices(1)

<table>
<thead>
<tr>
<th>NAME</th>
<th>list_devices – list allocatable devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNOPSIS</td>
<td>list_devices [ -s ] [-U uid] -l [device]</td>
</tr>
<tr>
<td></td>
<td>list_devices [ -s ] [-U uid] -n [device]</td>
</tr>
<tr>
<td></td>
<td>list_devices [ -s ] [-U uid] -u [device]</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>The <code>list_devices</code> utility lists the allocatable devices in the system according to specified qualifications. The <code>device</code> and all device special files associated with the device are listed. The device argument is optional and, if it is not present, all relevant devices are listed.</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>The following options are supported:</td>
</tr>
<tr>
<td></td>
<td><code>-l [device]</code> Lists the pathname(s) of the device special files associated with the device that are allocatable to the current process. If <code>device</code> is given, lists only the files associated with the specified device.</td>
</tr>
<tr>
<td></td>
<td><code>-n [device]</code> Lists the pathname(s) of device special files associated with the device that are allocatable to the current process but are not currently allocated. If <code>device</code> is given, lists only the files associated with that device.</td>
</tr>
<tr>
<td></td>
<td><code>-s</code> Silent. Suppresses any diagnostic output.</td>
</tr>
<tr>
<td></td>
<td><code>-u [device]</code> Lists the pathname(s) of device special files, associated with the device that are allocated to the owner of the current process. If <code>device</code> is given, list only the files associated with that device.</td>
</tr>
<tr>
<td></td>
<td><code>-U uid</code> Uses the user ID <code>uid</code> instead of the real user ID of the current process when performing the <code>list_devices</code> operation. Only a user with the <code>solaris.devices.revoke</code> authorization can use this option.</td>
</tr>
<tr>
<td>EXIT STATUS</td>
<td>The following exit values are returned:</td>
</tr>
<tr>
<td></td>
<td>non—zero An error occurred.</td>
</tr>
<tr>
<td>FILES</td>
<td>/etc/security/device_allocate</td>
</tr>
<tr>
<td></td>
<td>/etc/security/device_maps</td>
</tr>
<tr>
<td></td>
<td>/etc/security/dev/*</td>
</tr>
<tr>
<td></td>
<td>/usr/security/lib/*</td>
</tr>
</tbody>
</table>
| ATTRIBUTES   | See attributes(5) for descriptions of the following attributes:
### SEE ALSO

allocate(1), deallocate(1), bsmconv(1M), dminfo(1M), device_allocate(4),
device_maps(4), attributes(5)

### NOTES

The functionality described in this man page is available only if the Basic Security Module (BSM) has been enabled. See bsmconv(1M) for more information.
listusers(1)

NAME  listusers – list user login information

SYNOPSIS  listusers [-g groups] [-l logins]

DESCRIPTION  Executed without any options, this command lists all user logins sorted by login. The output shows the login ID and the account field value from the system’s password database as specified by /etc/nsswitch.conf.

OPTIONS  The following options are supported:

- `g groups`       Lists all user logins belonging to group, sorted by login. Multiple groups can be specified as a comma-separated list.
- `l logins`       Lists the user login or logins specified by logins, sorted by login. Multiple logins can be specified as a comma-separated list.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO  nsswitch.conf(4), attributes(5)

NOTES  A user login is one that has a UID of 100 or greater.

The `-l` and `-g` options can be combined. User logins will only be listed once, even if they belong to more than one of the selected groups.
llc2_autoconfig(1)

NAME llc2_autoconfig – generate LLC2 configuration files

SYNOPSIS /usr/lib/llc2/llc2_autoconfig [-f]

DESCRIPTION The llc2_autoconfig utility is used to generate LLC2 configuration files
(/etc/llc2/default/llc2.*). If there is no configuration file in
/etc/llc2_default/, it detects all the available interfaces in the system and
generates corresponding default configuration files.

If there are existing configuration files in /etc/llc2_default/, it will check if those
interfaces defined in the files still exist. If they do not exist in the system, it will set
llc2_on in those files to 0. After this, it will detect if there are new interfaces in the
system. If there are, it will generate configuration files for them.

OPTIONS The following option is supported:

- f Erases all configuration files in /etc/llc2/default/. Then detects all
the available interfaces in the system and generates corresponding default
configuration files. Use this option with caution.

FILES /etc/llc2/default/llc2.* LLC2 configuration files

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWllc</td>
</tr>
</tbody>
</table>

SEE ALSO llc2_config(1), llc2(4), attributes(5), llc2(7D)
NAME  llc2_config – configure LLC2 interface parameters

SYNOPSIS  /usr/lib/llc2/llc2_config [-P | -U | -d | -q | -i ppa | -r ppa]

DESCRIPTION  The llc2_config utility is used to start/stop the LLC2 subsystem and to configure LLC2 interface parameters.

OPTIONS  The following options are supported:

-\[\text{-d}\]  \hspace{1em} \text{Turns on debug mode. Extra debugging information will be printed out.}

-\[\text{-i ppa}\]  \hspace{1em} \text{Initializes the corresponding interface using the file}  
\hspace{1em} \text{/etc/llc2/default/llc2.ppa.}

-\[\text{-P}\]  \hspace{1em} \text{Reads in all /etc/llc2/default/llc2.* configuration files, opens}  
\hspace{1em} \text{those devices defined in the files, and sets up the streams needed for LLC2}  
\hspace{1em} \text{to use those devices. Before doing this, llc2_config-q will not show}  
\hspace{1em} \text{anything.}

-\[\text{-q}\]  \hspace{1em} \text{Queries the LLC2 subsystem. Information similar to the following example}  
\hspace{1em} \text{will be printed out for all PPAs (Physical Point of Attachment) available}  
\hspace{1em} \text{under the LLC2 module:}

<table>
<thead>
<tr>
<th>PPA</th>
<th>State</th>
<th>ID</th>
<th>MACAddr</th>
<th>Type</th>
<th>MaxSDU</th>
<th>MinSDU</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>up</td>
<td>0000</td>
<td>0800208a217e</td>
<td>ethernet</td>
<td>1500</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

The fields displayed are described below:

PPA  The relative logical position of the interface.
State The state of the interface:
up The interface is initialized and operational.
down The interface was "discovered" by the LLC2 driver, has passed its bootup diagnostics, and is awaiting initialization.
bad The interface is known to the LLC2 driver, but failed one or more of the integrity checks performed at boot time. This might include detecting Interrupt Request and shared memory conflicts or failures detected during the execution of the level 0 diagnostics.

ID The interface ID.
MACAddr The MAC address currently in effect for the interface.
Type The MAC type. Current types supported include:
llc2_config(1)

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWllc</td>
</tr>
</tbody>
</table>

SEE ALSO 11c2_autoconfig(1), 11c2(4), attributes(5), 11c2(7D)
The `llc2_stats` command is used to retrieve statistical information from the Host-based Logical Link Control Class 2 component of the LLC2 Driver. Statistics are kept for the station, SAP (Service Access Point), and connection components.

### OPTIONS
The following options are supported:

- `-c connection` Specifies the connection of interest. Its value is entered in hexadecimal notation with no leading 0x.
- `-r` Resets the specified counters to zero after reading them. This option is only valid if the `root` user is executing the command.
- `-s sap` Specifies the SAP for this request. It is a single-byte value, expressed in hexadecimal notation with no leading 0x. For example, the NetBIOS sap, 240 (0xf0) would be entered as: `-s f0`.

### OPERANDS
The following operand is supported:

- `ppa` The logical number used to address the adapter. The PPA (Physical Point of Attachment) must be the first argument.

### EXAMPLES
**EXAMPLE 1 Station Statistics**

The following command will display the station statistics for PPA 4. After the example, a brief description of each field is presented.

```bash
eexample% /usr/lib/llc2/llc2_stats 4
```

Station values received:

- `ppa` = 0x00000004
- `clearFlag` = 0x00
- `# of saps (hex)` = 0x0002
- `saps (hex)` = 02 aa
- `state` = 0x01
- `nullSapXidCmdRcvd` = 0x00000000
- `nullSapXidRspSent` = 0x00000000
- `nullSapTestCmdRcvd` = 0x00000000
- `nullSapTestRspSent` = 0x00000000
- `outOfState` = 0x00000000
- `allocFail` = 0x00000000
- `protocolError` = 0x00000000

The fields are described as follows:

- `ppa` The logical number used to address the adapter.
- `clearFlag` This flag indicates if the statistics will be reset to zero after reading (set to a 1) or if the statistics are read only (set to 0).
EXAMPLE 1 Station Statistics  (Continued)

# of saps  The number of SAPs currently bound on this station.
saps  The array of the station’s Service Access Point (SAP) logical interface values between the LLC and its adjacent layers.
state  A number indicating the current state of the station component (0 = down, 1 = up).
nullSapXidCmdRcvd  The number of XID command Protocol Data Units (PDUs) received for the NULL SAP address (sap = 0x00).
nullSapXidRspSent  The number of XID response PDUs sent in response to XID command PDUs received for the null SAP address.
nullSapTestCmdRcvd  The number of TEST command PDUs received for the null SAP address.
nullSapTestRspSent  The number of TEST response PDUs sent in response to TEST command PDUs received for the null SAP address.
outOfState  The number of events received in an invalid state.
allocFail  The number of buffer allocation failures.
protocolError  The number of LLC protocol errors, that is, the receipt of malformed PDUs or the receipt of frame X when frame Y was expected.

EXAMPLE 2 SAP Statistics

In the above display, there are two active SAPs, 0x02 and 0xaa. The following is an example of a command for retrieving the statistics for SAP 02 and a brief explanation of each field presented.

eexample% /usr/lib/llc2/llc2_stats 4 -s 02

Sap values received:
ppa = 0x00000004  clearFlag = 0x00
sap = 0x02
state = 0x01
# of cons (hex) = 0x0000000a
connections (hex) = 0000 0001 0002 0003 0004 0005 0006 0007 0008 0009
xidCmdSent = 0x00000000
xidCmdRcvd = 0x00000000
xidRspSent = 0x00000000
xidRspRcvd = 0x00000000
testCmdSent = 0x00000000
testCmdRcvd = 0x00000000
testRspSent = 0x00000000
EXAMPLE 2 SAP Statistics  (Continued)

testRspRcvd = 0x00000000
uiSent = 0x00000000
uiRcvd = 0x00000000
outOfState = 0x00000000
allocFail = 0x00000000
protocolError = 0x00000000

The fields are described as follows:

ppa The logical number used to address the adapter.
clearFlag This flag indicates if the statistics will be reset to zero after reading (set to 1) or if the statistics are read only (set to 0).
sap The specified Service Access Point (SAP) logical interface value for the station.
state A number indicating the current state of the SAP component (0 = inactive, 1 = active).
# of cons The number of active connections on this SAP.
connections The array of active connection indexes.
xidCmdSent The number of XID command PDUs sent (Source SAP = this sap).
xidCmdRcvd The number of XID command PDUs received (Destination SAP = this sap).
xidRspSent The number of XID response PDUs sent (Source SAP = this sap).
xidRspRcvd The number of XID response PDUs received (Source SAP = this sap).
testCmdSent The number of TEST command PDUs sent (Source SAP = this sap).
testCmdRcvd The number of TEST command PDUs received (Destination SAP = this sap).
testRspSent The number of TEST response PDUs sent (Source SAP = this sap).
testRspRcvd The number of TEST response PDUs received (Source SAP = this sap).
uiSent The number of Unnumbered Information Frames sent.
uiRcvd The number of Unnumbered Information Frames received.
EXAMPLE 2 SAP Statistics (Continued)

outOfState    The number of events received in an invalid state.
allocFail     The number of buffer allocation failures.
protocolError The number of LLC protocol errors, that is, the receipt of malformed PDUs or the receipt of frame X when frame Y was expected.

EXAMPLE 3 Connection Statistics

Ten established connections are associated with this SAP. To retrieve the statistics for connection 1, enter the following command:

```
example$ /usr/lib/llc2/llc2_stats 4 -s 2 -c 1
```

Connection values received:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ppa</td>
<td>0x0004</td>
</tr>
<tr>
<td>clearFlag</td>
<td>0x00</td>
</tr>
<tr>
<td>sap</td>
<td>0x02</td>
</tr>
<tr>
<td>con</td>
<td>0x0001</td>
</tr>
<tr>
<td>sid</td>
<td>0x0201</td>
</tr>
<tr>
<td>stateOldest</td>
<td>0x00</td>
</tr>
<tr>
<td>stateOlder</td>
<td>0x00</td>
</tr>
<tr>
<td>state</td>
<td>0x08</td>
</tr>
<tr>
<td>d1_nodeaddr</td>
<td>0x0080d84008c2</td>
</tr>
<tr>
<td>d1_sap</td>
<td>0x04</td>
</tr>
<tr>
<td>flag</td>
<td>0x50</td>
</tr>
<tr>
<td>dataFlag</td>
<td>0x00</td>
</tr>
<tr>
<td>timerOn</td>
<td>0x10</td>
</tr>
<tr>
<td>vs</td>
<td>0x29</td>
</tr>
<tr>
<td>vr</td>
<td>0x1e</td>
</tr>
<tr>
<td>mrRcvd</td>
<td>0x29</td>
</tr>
<tr>
<td>k</td>
<td>0x14</td>
</tr>
<tr>
<td>retryCount</td>
<td>0x0000</td>
</tr>
<tr>
<td>numToBeAcked</td>
<td>0x0000</td>
</tr>
<tr>
<td>numToResend</td>
<td>0x0000</td>
</tr>
<tr>
<td>macOutSave</td>
<td>0x0000</td>
</tr>
<tr>
<td>macOutDump</td>
<td>0x0000</td>
</tr>
<tr>
<td>iSent</td>
<td>0x0ba9</td>
</tr>
<tr>
<td>iRcvd</td>
<td>0x0001e</td>
</tr>
<tr>
<td>frmrrSent</td>
<td>0x00000</td>
</tr>
<tr>
<td>frmrrRcvd</td>
<td>0x00000</td>
</tr>
<tr>
<td>rrSent</td>
<td>0x016a</td>
</tr>
<tr>
<td>rrRcvd</td>
<td>0x00c1</td>
</tr>
<tr>
<td>rnrSent</td>
<td>0x0000</td>
</tr>
<tr>
<td>rnrRcvd</td>
<td>0x06fb</td>
</tr>
<tr>
<td>rejSent</td>
<td>0x0000</td>
</tr>
<tr>
<td>rejRcvd</td>
<td>0x0000</td>
</tr>
<tr>
<td>sabmeSent</td>
<td>0x0000</td>
</tr>
<tr>
<td>sabmeRcvd</td>
<td>0x0001</td>
</tr>
<tr>
<td>uaSent</td>
<td>0x0001</td>
</tr>
<tr>
<td>uaRcvd</td>
<td>0x0000</td>
</tr>
<tr>
<td>discSent</td>
<td>0x0000</td>
</tr>
<tr>
<td>outOfState</td>
<td>0x0000</td>
</tr>
<tr>
<td>allocFail</td>
<td>0x0000</td>
</tr>
<tr>
<td>protocolError</td>
<td>0x0000</td>
</tr>
<tr>
<td>localBusy</td>
<td>0x0000</td>
</tr>
<tr>
<td>remoteBusy</td>
<td>0x00b5</td>
</tr>
<tr>
<td>maxRetryFail</td>
<td>0x0000</td>
</tr>
<tr>
<td>ackTimerExp</td>
<td>0x0000</td>
</tr>
<tr>
<td>pollTimerExp</td>
<td>0x0000</td>
</tr>
<tr>
<td>rejTimerExp</td>
<td>0x0000</td>
</tr>
<tr>
<td>remBusyTimerExp</td>
<td>0x0000</td>
</tr>
<tr>
<td>inactTimerExp</td>
<td>0x0000</td>
</tr>
<tr>
<td>sendAckTimerExp</td>
<td>0x0000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ppa</td>
<td></td>
</tr>
<tr>
<td>clearFlag</td>
<td></td>
</tr>
<tr>
<td>sap</td>
<td></td>
</tr>
<tr>
<td>con</td>
<td></td>
</tr>
<tr>
<td>stateOldest</td>
<td></td>
</tr>
</tbody>
</table>
### EXAMPLE 3 Connection Statistics  (Continued)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stateOlder</td>
<td>A number representing the state of the connection component prior to stateOld.</td>
</tr>
<tr>
<td>stateOld</td>
<td>A number representing the state of the connection component prior to state.</td>
</tr>
<tr>
<td>state</td>
<td>A number representing the most current state of the connection component. See Table 1.</td>
</tr>
<tr>
<td>sid</td>
<td>The Station Identifier composed of the SAP (upper byte) and connection index (lower byte).</td>
</tr>
<tr>
<td>dl_nodeaddr</td>
<td>The Data Link Node Address. This is the destination node’s MAC address.</td>
</tr>
<tr>
<td>dl_sap</td>
<td>The destination node’s SAP.</td>
</tr>
<tr>
<td>flag</td>
<td>The connection component processing flag. See Table 3.</td>
</tr>
<tr>
<td>dataFlag</td>
<td>A number representing the status of the data units from received I-frame PDUs (0 = not discarded, 1 = discarded, 2 = busy state entered with REJ PDU outstanding).</td>
</tr>
<tr>
<td>timerOn</td>
<td>A number representing the timer activity flag, with each bit representing an active timer for this connection. See Table 2 for timer definitions.</td>
</tr>
<tr>
<td>vs</td>
<td>The sequence number of the next I-frame PDU to send.</td>
</tr>
<tr>
<td>vr</td>
<td>The expected sequence number of the next I-frame PDU to be received.</td>
</tr>
<tr>
<td>nrRcvd</td>
<td>The sequence number plus 1 of the last sent I-frame PDU acknowledged by the remote node.</td>
</tr>
<tr>
<td>k</td>
<td>The transmit window size.</td>
</tr>
<tr>
<td>retryCount</td>
<td>The retryCount is incremented whenever a timer expiration occurs. These timers protect outbound frames.</td>
</tr>
<tr>
<td>numToBeAched</td>
<td>The number of outbound I-frames awaiting acknowledgement.</td>
</tr>
<tr>
<td>numToResend</td>
<td>The number of outbound I-frames to be retransmitted.</td>
</tr>
<tr>
<td>macOutSave</td>
<td>No longer used.</td>
</tr>
<tr>
<td>macOutDump</td>
<td>No longer used.</td>
</tr>
<tr>
<td>iSent</td>
<td>The number of I-frames sent.</td>
</tr>
</tbody>
</table>
### EXAMPLE 3 Connection Statistics (Continued)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iRcvd</td>
<td>The number of I-frames received.</td>
</tr>
<tr>
<td>frmrSent</td>
<td>The number of Frame Reject PDUs (FRMR) sent.</td>
</tr>
<tr>
<td>frmrRcvd</td>
<td>The number of Frame Reject PDUs (FRMR) received.</td>
</tr>
<tr>
<td>rrSent</td>
<td>The number of Receiver Ready PDUs (RR) sent.</td>
</tr>
<tr>
<td>rrRcvd</td>
<td>The number of Receiver Ready PDUs (RR) received.</td>
</tr>
<tr>
<td>rnrSent</td>
<td>The number of Receiver Not Ready PDUs (RNR) sent.</td>
</tr>
<tr>
<td>rnrRcvd</td>
<td>The number of Receiver Not Ready PDUs (RNR) received.</td>
</tr>
<tr>
<td>rejSent</td>
<td>The number of Reject PDUs (REJ) sent.</td>
</tr>
<tr>
<td>rejRcvd</td>
<td>The number of Reject PDUs (REJ) received.</td>
</tr>
<tr>
<td>sabmeSent</td>
<td>The number of Set Asynchronous Balanced Mode Extended PDUs (SABME) sent.</td>
</tr>
<tr>
<td>sabmeRcvd</td>
<td>The number of Set Asynchronous Balanced Mode Extended PDUs (SABME) received.</td>
</tr>
<tr>
<td>uaSent</td>
<td>The number of Unnumbered Acknowledgment PDUs (UA) sent.</td>
</tr>
<tr>
<td>uaRcvd</td>
<td>The number of Unnumbered Acknowledgment PDUs (UA) received.</td>
</tr>
<tr>
<td>discSent</td>
<td>The number of Disconnect PDUs (DISC) sent.</td>
</tr>
<tr>
<td>outOfState</td>
<td>The number of events received in an invalid state.</td>
</tr>
<tr>
<td>allocFail</td>
<td>The number of buffer allocation failures.</td>
</tr>
<tr>
<td>protocolError</td>
<td>The number of LLC protocol errors, that is, the receipt of malformed PDUs or the receipt of frame X when frame Y was expected.</td>
</tr>
<tr>
<td>localBusy</td>
<td>The number of times this component was in local busy state and could not accept I-frames.</td>
</tr>
<tr>
<td>remoteBusy</td>
<td>The number of times the remote connection component was busy and could not accept I-frames.</td>
</tr>
<tr>
<td>maxRetryFail</td>
<td>The number of failures that occurred because maxRetry was reached.</td>
</tr>
<tr>
<td>ackTimerExp</td>
<td>The number of expirations of the Acknowledgement timer.</td>
</tr>
<tr>
<td>pollTimerExp</td>
<td>The number of expirations of the Poll timer.</td>
</tr>
</tbody>
</table>
EXAMPLE 3 Connection Statistics  (Continued)

<table>
<thead>
<tr>
<th>Timer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rejTimerExp</td>
<td>The number of expirations of the Reject timer.</td>
</tr>
<tr>
<td>remBusyTimerExp</td>
<td>The number of expirations of the Remote Busy timer.</td>
</tr>
<tr>
<td>inactTimerExp</td>
<td>The number of expirations of the Inactivity timer.</td>
</tr>
<tr>
<td>sendAckTimerExp</td>
<td>The number of expirations of the Send Acknowledgement timer.</td>
</tr>
</tbody>
</table>

**Table 1: LLC2 States**

<table>
<thead>
<tr>
<th>STATION</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>~DOWN</td>
<td>0x00</td>
</tr>
<tr>
<td>~UP</td>
<td>0x01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SAP</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>~INACTIVE</td>
<td>0x00</td>
</tr>
<tr>
<td>~ACTIVE</td>
<td>0x01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CONNECTION</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>~ADM</td>
<td>0x00</td>
</tr>
<tr>
<td>~CONN</td>
<td>0x01</td>
</tr>
<tr>
<td>~RESET_WAIT</td>
<td>0x02</td>
</tr>
<tr>
<td>~RESET_CHECK</td>
<td>0x03</td>
</tr>
<tr>
<td>~SETUP</td>
<td>0x04</td>
</tr>
<tr>
<td>~RESET</td>
<td>0x05</td>
</tr>
<tr>
<td>~D_CONN</td>
<td>0x06</td>
</tr>
<tr>
<td>~ERROR</td>
<td>0x07</td>
</tr>
<tr>
<td>~NORMAL</td>
<td>0x08</td>
</tr>
<tr>
<td>~BUSY</td>
<td>0x09</td>
</tr>
<tr>
<td>~REJECT</td>
<td>0x0a</td>
</tr>
<tr>
<td>~AWAIT</td>
<td>0x0b</td>
</tr>
<tr>
<td>~AWAIT_BUSY</td>
<td>0x0c</td>
</tr>
<tr>
<td>~AWAIT_REJECT</td>
<td>0x0d</td>
</tr>
</tbody>
</table>
llc2_stats(1)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWllc</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Attribute Type</th>
<th>Attribute Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWllc</td>
</tr>
</tbody>
</table>

FILES
/dev/llc2
clone device

SEE ALSO
attributes(5)

NOTES
For further information on the LLC2 components, states and flags, see the International Standards Organization document, ISO 8802-2: 1994, Section 7.
NAME

ln - make hard or symbolic links to files

SYNOPSIS

/usr/bin/ln [-fns] source_file [target]
/usr/bin/ln [-fns] source_file... target
/usr/xpg4/bin/ln [-fs] source_file [target]
/usr/xpg/bin/ln [-fs] source_file... target

DESCRIPTION

In the first synopsis form, the ln utility creates a new directory entry (link) for the file specified by source_file, at the destination path specified by target. If target is not specified, the link is made in the current directory. This first synopsis form is assumed when the final operand does not name an existing directory; if more than two operands are specified and the final is not an existing directory, an error will result.

In the second synopsis form, the ln utility creates a new directory entry for each file specified by a source_file operand, at a destination path in the existing directory named by target.

The ln utility may be used to create both hard links and symbolic links. A hard link is a pointer to a file and is indistinguishable from the original directory entry. Any changes to a file are effective independent of the name used to reference the file. Hard links may not span file systems and may not refer to directories.

ln by default creates hard links. source_file is linked to target. If target is a directory, another file named source_file is created in target and linked to the original source_file.

/usr/bin/ln
If target is a file, its contents are overwritten. If /usr/bin/ln determines that the mode of target forbids writing, it will print the mode (see chmod(1)), ask for a response, and read the standard input for one line. If the response is affirmative, the link occurs, if permissible; otherwise, the command exits.

/usr/xpg4/bin/ln
If target is a file and the -f option is not specified, /usr/xpg4/bin/ln will write a diagnostic message to standard error, do nothing more with the current source_file, and go on to any remaining source_files.

A symbolic link is an indirect pointer to a file; its directory entry contains the name of the file to which it is linked. Symbolic links may span file systems and may refer to directories.

When creating a hard link, and the source file is itself a symbolic link, then the target will be a hard link to the file referenced by the symbolic link, not to the symbolic link object itself (source_file).

File permissions for target may be different from those displayed with a -l listing of the ls(1) command. To display the permissions of target use ls -ll. See stat(2) for more information.

OPTIONS

The following options are supported for both /usr/bin/ln and /usr/xpg4/bin/ln:
-f  Link files without questioning the user, even if the mode of target forbids writing. This is the default if the standard input is not a terminal.

-s  Create a symbolic link.

If the -s option is used with two arguments, target may be an existing directory or a non-existent file. If target already exists and is not a directory, an error is returned. source_file may be any path name and need not exist. If it exists, it may be a file or directory and may reside on a different file system from target. If target is an existing directory, a file is created in directory target whose name is source_file or the last component of source_file. This file is a symbolic link that references source_file. If target does not exist, a file with name target is created and it is a symbolic link that references source_file.

If the -s option is used with more than two arguments, target must be an existing directory or an error will be returned. For each source_file, a link is created in target whose name is the last component of source_file; each new source_file is a symbolic link to the original source_file. The files and target may reside on different file systems.

The following options are supported for /usr/bin/ln only:

-n  If the link is an existing file, do not overwrite the contents of the file. The
    -f option overrides this option. This is the default behavior for
    /usr/xpg4/bin/ln, and is silently ignored.

OPERANDS

The following operands are supported:

source_file  A path name of a file to be linked. This can be either a regular or
             special file. If the -s option is specified, source_file can also be a
directory.

target  The path name of the new directory entry to be created, or of an
         existing directory in which the new directory entries are to be
         created.

USAGE

See largefile(5) for the description of the behavior of ln when encountering files
greater than or equal to 2 Gbyte (2^31 bytes).

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the
execution of ln: LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS

The following exit values are returned:

0  All the specified files were linked successfully

>0  An error occurred.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:
A symbolic link to a directory behaves differently than you might expect in certain cases. While an `ls(1)` on such a link displays the files in the pointed-to directory, an `'ls -l'` displays information about the link itself:

```
example% ln -s dir link
example% ls link
file1 file2 file3 file4
example% ls -l link
lrwxrwxrwx 1 user 7 Jan 11 23:27 link -> dir
```

When you `cd(1)` to a directory through a symbolic link, you wind up in the pointed-to location within the file system. This means that the parent of the new working directory is not the parent of the symbolic link, but rather, the parent of the pointed-to directory. For instance, in the following case the final working directory is `/usr` and not `/home/user/linktest`.

```
example% pwd
/home/user/linktest
example% ln -s /usr/tmp symlink
example% cd symlink
example% cd ..
example% pwd
/usr
```

C shell users can avoid any resulting navigation problems by using the `pushd` and `popd` built-in commands instead of `cd`.

SEE ALSO

`chmod(1)`, `ls(1)`, `stat(2)`, `attributes(5)`, `environ(5)`, `largefile(5)`, `XPG4(5)`

NOTES

A symbolic link to a directory behaves differently than you might expect in certain cases. While an `ls(1)` on such a link displays the files in the pointed-to directory, an `'ls -l'` displays information about the link itself:
ln(1B)

NAME
ln – make hard or symbolic links to files

SYNOPSIS
/usr/ucb/ln [-fs] filename [linkname]
/usr/ucb/ln [-fs] pathname... directory

DESCRIPTION
The /usr/ucb/ln utility creates an additional directory entry, called a link, to a file or directory. Any number of links can be assigned to a file. The number of links does not affect other file attributes such as size, protections, data, etc.

filename is the name of the original file or directory. linkname is the new name to associate with the file or filename. If linkname is omitted, the last component of filename is used as the name of the link.

If the last argument is the name of a directory, symbolic links are made in that directory for each pathname argument; /usr/ucb/ln uses the last component of each pathname as the name of each link in the named directory.

A hard link (the default) is a standard directory entry just like the one made when the file was created. Hard links can only be made to existing files. Hard links cannot be made across file systems (disk partitions, mounted file systems). To remove a file, all hard links to it must be removed, including the name by which it was first created; removing the last hard link releases the inode associated with the file.

A symbolic link, made with the -s option, is a special directory entry that points to another named file. Symbolic links can span file systems and point to directories. In fact, you can create a symbolic link that points to a file that is currently absent from the file system; removing the file that it points to does not affect or alter the symbolic link itself.

A symbolic link to a directory behaves differently than you might expect in certain cases. While an ls(1) on such a link displays the files in the pointed-to directory, an `ls -l` displays information about the link itself:

```
example% /usr/ucb/ln -s dir link
example% ls link
file1 file2 file3 file4
example% ls -l link
lrwxrwxrwx 1 user 7 Jan 11 23:27 link → dir
```

When you use cd(1) to change to a directory through a symbolic link, you wind up in the pointed-to location within the file system. This means that the parent of the new working directory is not the parent of the symbolic link, but rather, the parent of the pointed-to directory. For instance, in the following case the final working directory is /usr and not /home/user/linktest.

```
exa...
C shell user’s can avoid any resulting navigation problems by using the pushd and popd built-in commands instead of cd.

**OPTIONS**

- 
  Force a hard link to a directory. This option is only available to the super-user, and should be used with extreme caution.

- 
  Create a symbolic link or links.

**USAGE**

See `largefile(5)` for the description of the behavior of `ln` when encountering files greater than or equal to 2 Gbyte (231 bytes).

**EXAMPLES**

**EXAMPLE 1 The /usr/ucb/ln command**

The commands below illustrate the effects of the different forms of the `/usr/ucb/ln` command:

```
example$ /usr/ucb/ln file link
example$ ls -F file link
file link
example$ /usr/ucb/ln -s file symlink
example$ ls -F file symlink
file symlink@
example$ ls -li file link symlink
10606 -rw-r--r-- 2 user 0 Jan 12 00:06 file
10606 -rw-r--r-- 2 user 0 Jan 12 00:06 link
10607 lrwxrwxrwx 1 user 4 Jan 12 00:06 symlink → file
example$ /usr/ucb/ln -s nonesuch devoid
example$ ls -F devoid
devoid@
example$ cat devoid
devoid: No such file or directory
example$ /usr/ucb/ln -s /proto/bin/* /tmp/bin
example$ ls -F /proto/bin /tmp/bin
/proto/bin:
x*  y*  z*
/tmp/bin:
x*  y*  z*
```

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

**SEE ALSO**
cp(1), ls(1), mv(1), rm(1), link(2), readlink(2), stat(2), symlink(2), attributes(5), largefile(5)
When the last argument is a directory, simple basenames should not be used for \textit{pathname} arguments. If a basename is used, the resulting symbolic link points to itself:

```bash
example$ /usr/ucb/ln -s file /tmp
example$ ls -l /tmp/file
lrwxrwxrwx 1 user 4 Jan 12 00:16 /tmp/file → file
example$ cat /tmp/file
/tmp/file: Too many levels of symbolic links
```

To avoid this problem, use full pathnames, or prepend a reference to the PWD variable to files in the working directory:

```bash
example$ rm /tmp/file
example$ /usr/ucb/ln -s $PWD/file /tmp
lrwxrwxrwx 1 user 4 Jan 12 00:16 /tmp/file → 
/home/user/subdir/file
```
NAME
loadkeys, dumpkeys – load and dump keyboard translation tables

SYNOPSIS
loadkeys [filename]
dumpkeys

DESCRIPTION
loadkeys reads the file specified by filename, and modifies the keyboard streams module’s translation tables. If no file is specified, loadkeys loads the file:
/usr/share/lib/keytables/type_tt/layout_dd, where tt is the value returned by the KIOCTYPE ioctl, and dd is the value returned by the KIOCLAYOUT ioctl (see kb(7M)). These keytable files specify only the entries that change between the specified layout and the default layout for the particular keyboard type. On self-identifying keyboards, the value returned by the KIOCLAYOUT ioctl is set from the DIP switches.

dumpkeys writes the current contents of the keyboard streams module’s translation tables, in the format specified by keytables(4), to the standard output.

FILES
/usr/share/lib/keytables/layout_dd
default keytable files

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Evolving</td>
</tr>
</tbody>
</table>

SEE ALSO
kbd(1), keytables(4), attributes(5), kb(7M), usbkbm(7M)
locale(1)

NAME locale – get locale-specific information

SYNOPSIS locale [-a | -m]
locale [-ck] name...

DESCRIPTION The locale utility writes information about the current locale environment, or all
public locales, to the standard output. For the purposes of this section, a public locale is
one provided by the implementation that is accessible to the application.

When locale is invoked without any arguments, it summarizes the current locale
environment for each locale category as determined by the settings of the environment
variables.

When invoked with operands, it writes values that have been assigned to the
keywords in the locale categories, as follows:

- Specifying a keyword name selects the named keyword and the category
  containing that keyword.
- Specifying a category name selects the named category and all keywords in that
category.

OPTIONS The following options are supported:

- `a` Write information about all available public locales. The available locales
  include POSIX, representing the POSIX locale.
- `c` Write the names of selected locale categories. The -c option increases
  readability when more than one category is selected (for example, via more
  than one keyword name or via a category name). It is valid both with and
  without the -k option.
- `k` Write the names and values of selected keywords. The implementation
  may omit values for some keywords; see OPERANDS.
- `m` Write names of available charmaps; see localedef(1).

OPERANDS The following operand is supported:

name The name of a locale category, the name of a keyword in a locale category,
or the reserved name charmap. The named category or keyword will be
selected for output. If a single name represents both a locale category name
and a keyword name in the current locale, the results are unspecified;
otherwise, both category and keyword names can be specified as name
operands, in any sequence.

EXAMPLES EXAMPLE 1 Examples of the locale utility.

In the following examples, the assumption is that locale environment variables are set
as follows:

LANG=locale_x LC_COLLATE=locale_y
EXAMPLE 1  Examples of the locale utility.  (Continued)

The command locale would result in the following output:

```
LANG=locale_x
LC_CTYPE="locale_x"
LC_NUMERIC="locale_x"
LC_TIME="locale_x"
LC_COLLATE=locale_y
LC_MONETARY="locale_x"
LC_MESSAGES="locale_x"
LC_ALL=
```

The command `LC_ALL=POSIX locale -ck decimal_point` would produce:

```
LC_NUMERIC
decimal_point="."
```

The following command shows an application of `locale` to determine whether a user-supplied response is affirmative:

```
if printf "%s
" "$response" | /usr/xpg4/bin/grep -Eq "$(locale yesexpr)"
then
    affirmative processing goes here
else
    non-affirmative processing goes here
fi
```

**ENVIRONMENT VARIABLES**

See `environ(5)` for the descriptions of `LANG`, `LC_ALL`, `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

The `LANG`, `LC_*`, and `NLSPATH` environment variables must specify the current locale environment to be written out; they will be used if the `-a` option is not specified.

**EXIT STATUS**

The following exit values are returned:

- `0`  All the requested information was found and output successfully.
- `>0`  An error occurred.

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWloc</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

**SEE ALSO**

`localedef(1)`, `attributes(5)`, `charmap(5)`, `environ(5)`, `locale(5)`
NOTES

If LC_CTYPE or keywords in the category LC_CTYPE are specified, only the values in the range 0x00-0x7f are written out.

If LC_COLLATE or keywords in the category LC_COLLATE are specified, no actual values are written out.
localedef(1)

NAME
localedef – define locale environment

SYNOPSIS
localedef [-c] [-C compiler_options] [-f charmap] [-i sourcefile]
            [-L linker_options] [-m model] [-W cc, arg] [-x extensions_file] localename

DESCRIPTION
The localedef utility converts source definitions for locale categories into a format
usable by the functions and utilities whose operational behavior is determined by the
setting of the locale environment variables; see environ(5).

The utility reads source definitions for one or more locale categories belonging to the
same locale from the file named in the -i option (if specified) or from standard input.

Each category source definition is identified by the corresponding environment
variable name and terminated by an END category-name statement. The following
categories are supported.

  LC_CTYPE       Defines character classification and case conversion.
  LC_COLLATE     Defines collation rules.
  LC_MONETARY    Defines the format and symbols used in formatting of
                 monetary information.
  LC_NUMERIC    Defines the decimal delimiter, grouping and grouping
                 symbol for non-monetary numeric editing.
  LC_TIME       Defines the format and content of date and time
                 information.
  LC_MESSAGES   Defines the format and values of affirmative and
                 negative responses.

OPTIONS
The following options are supported:

  -c            Creates permanent output even if warning messages
                 have been issued.
  -C compiler_options
                 Passes the compiler_options to the C compiler (cc). If
                 more than one option is specified, then the options
                 must be enclosed in quotes (" ").
                 This is an old option. Use the -Wcc, arg option instead.
  -f charmap
                 Specifies the pathname of a file containing a mapping
                 of character symbols and collating element symbols to
                 actual character encodings. This option must be
                 specified if symbolic names (other than collating
                 symbols defined in a collating-symbol keyword)
                 are used. If the -f option is not present, the default
                 character mapping will be used.
localedef(1)

- i sourcefile
  The path name of a file containing the source definitions. If this option is not present, source definitions will be read from standard input.

- L linker_options
  Passes the linker_options to the C compiler (cc) that follows the C source filename. If more than one option is specified, then the options must be enclosed in quotes (" ").

  This is an old option. Use the -W cc, arg option instead.

- m model
  Specifies whether localedef will generate a 64-bit or a 32-bit locale object.

  Specify model as ilp32 to generate a 32-bit locale object. Specify lp64 to generate a 64-bit locale object. If the -m option is not specified, localedef generates a 32-bit locale object. And if no other options than -c, -f, and -i options are specified and if the system running localedef supports the 64-bit environment, localedef additionally generates a 64-bit locale object.

- W cc, arg
  Passes arg options to the C compiler. Each argument must be separated from the preceding by only a comma. (A comma can be part of an argument by escaping it by an immediately preceding backslash character; the backslash is removed from the resulting argument.)

  Use this option instead of the -C and -L options.

- x extensions_file
  Specifies the name of an extension file where various localedef options are listed. See locale(5).

OPERANDS
  The following operand is supported:

localename
  Identifies the locale. If the name contains one or more slash characters, localename will be interpreted as a path name where the created locale definitions will be stored. This capability may be restricted to users with appropriate privileges. (As a consequence of specifying one localename, although several categories can be processed in one execution, only categories belonging to the same locale can be processed.)

OUTPUT
  localedef creates a temporary C source file that represents the locale’s data.
  localedef then calls the C compiler to compile this C source file into a shared object.
localedef(1)

If the `-milp32` option is specified, `localedef` calls the C compiler for generating 32-bit objects and generates a 32-bit locale object. If the `-mlp64` option is specified, `localedef` calls the C compiler for generating 64-bit objects and generates a 64-bit locale object.

If the `-m` option is not specified, `localedef` calls the C compiler for generating 32-bit objects and generates a 32-bit locale object. If no other options than `-c`, `-f`, and `-i` options are specified and if the system running `localedef` supports the 64-bit environment, `localedef` additionally calls the C compiler for generating 64-bit objects and generates a 64-bit locale object.

If no option to the C compiler is explicitly specified using the `-W`, `-C`, or `-L` options, `localedef` calls the C compiler with appropriate C compiler options to generate a locale object or objects.

If the `-milp32` option is specified, `localedef` generates a 32-bit locale object named:

`localename.so.version_number`

If the `-mlp64` option is specified, `localedef` generates a 64-bit locale object named:

`localename.so.version_number`

If the `-m` option is not specified, `localedef` generates a 32-bit locale object named:

`localename.so.version_number`

and, if appropriate, generates a 64-bit locale object named:

`64-bit_architecture_name/localename.so.version_number`

The shared object for the 32-bit environment must be moved to:

`/usr/lib/locale/localename/localename.so.version_number`

The shared object for the 64-bit environment on SPARC must be moved to:

`/usr/lib/locale/localename/sparcv9/localename.so.version_number`

`localedef` also generates a text file named `localename` that is used for information only.

See `environ(5)` for definitions of the following environment variables that affect the execution of `localedef`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

User Commands  731
localedef(1)

**EXIT STATUS**

The following exit values are returned:

0    No errors occurred and the locales were successfully created.
1    Warnings occurred and the locales were successfully created.
2    The locale specification exceeded implementation limits or the coded character set or sets used were not supported by the implementation, and no locale was created.
3    The capability to create new locales is not supported by the implementation.
>3   Warnings or errors occurred and no output was created.

If an error is detected, no permanent output will be created.

**FILES**

/usr/lib/localedef/extensions/generic_eucbc.x
    Describes what a generic EUC locale uses in the system. This file is used by default.

/usr/lib/localedef/extensions/single_byte.x
    Describes a generic single-byte file used in the system.

/usr/lib/locale/localename/localename.so.version_number
    The shared object for the 32-bit environment.

/usr/lib/locale/localename/sparcv9/localename.so.version_number
    The shared object for the 64-bit environment on SPARC.

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

**SEE ALSO**

locale(1), nl_langinfo(3C), strftime(3C), attributes(5), charmap(5), environ(5), extensions(5), locale(5)

**WARNINGS**

If warnings occur, permanent output will be created if the -c option was specified.

The following conditions will cause warning messages to be issued:

- If a symbolic name not found in the charmap file is used for the descriptions of the LC_CTYPE or LC_COLLATE categories (for other categories, this will be an error condition).
- If optional keywords not supported by the implementation are present in the source.
logger(1)

NAME  logger – add entries to the system log

SYNOPSIS  logger [-i] [-f file] [-p priority] [-t tag] [message] ...

DESCRIPTION  The logger command provides a method for adding one-line entries to the system log file from the command line. One or more message arguments can be given on the command line, in which case each is logged immediately. If this is unspecified, either the file indicated with -f or the standard input is added to the log. Otherwise, a file can be specified, in which case each line in the file is logged. If neither is specified, logger reads and logs messages on a line-by-line basis from the standard input.

OPTIONS  The following options are supported:

-ffile  Use the contents of file as the message to log.

-i  Log the process ID of the logger process with each line.

-ppriority  Enter the message with the specified priority. The message priority can be specified numerically, or as a facility.level pair. For example, '-p local3.info' assigns the message priority to the info level in the local3 facility. The default priority is user.notice.

-ttag  Mark each line added to the log with the specified tag.

OPERANDS  The following operand is supported:

message  One of the string arguments whose contents are concatenated together, in the order specified, separated by single space characters.

EXAMPLES  EXAMPLE 1 Examples of the logger command.

The following example:

example$ logger System rebooted

logs the message ‘System rebooted’ to the default priority level notice to be treated by syslogd as are other messages to the facility user.

The next example:

example$ logger -p local0.notice -t HOSTIDM -f /dev/idmc

reads from the file /dev/idmc and logs each line in that file as a message with the tag ‘HOSTIDM’ at priority level notice to be treated by syslogd as are other messages to the facility local0.

ENVIRONMENT VARIABLES  See environ(5) for descriptions of the following environment variables that affect the execution of logger: LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS  The following exit values are returned:

0  Successful completion.
An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO mailx(1), write(1), syslogd(1M), syslog(3C), attributes(5), environ(5)
The `logger` utility provides a method for adding one-line entries to the system log file from the command line. One or more `message` arguments can be given on the command line, in which case each is logged immediately. If `message` is unspecified, either the file indicated with `-f` or the standard input is added to the log. Otherwise, a `filename` can be specified, in which case each line in the file is logged. If neither is specified, `logger` reads and logs messages on a line-by-line basis from the standard input.

**OPTIONS**

The following options are supported:

- `-i` Log the process ID of the `logger` process with each line.
- `-f filename` Use the contents of `filename` as the message to log.
- `-p priority` Enter the message with the specified `priority`. The message priority can be specified numerically, or as a `facility.level` pair. For example, `-p local3.info` assigns the message priority to the info level in the local3 facility. The default priority is `user.notice`.
- `-t tag` Mark each line added to the log with the specified `tag`.

**EXAMPLES**

**EXAMPLE 1** Logging a message

The command:
```
example% logger System rebooted
```
will log the message ‘System rebooted’ to the facility at priority `notice` to be treated by `syslogd` as other messages to the facility `notice` are.

**EXAMPLE 2** Logging messages from a file

The command:
```
example% logger -p local0.notice -t HOSTIDM -f /dev/idmc
```
will read from the file `/dev/idmc` and will log each line in that file as a message with the tag ‘HOSTIDM’ at priority `notice` to be treated by `syslogd` as other messages to the facility `local0` are.

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpuc</td>
</tr>
</tbody>
</table>
SEE ALSO syslogd(1M), syslog(3C), attributes(5)
The `login` command is used at the beginning of each terminal session to identify oneself to the system. `login` is invoked by the system when a connection is first established, after the previous user has terminated the login shell by issuing the `exit` command.

If `login` is invoked as a command, it must replace the initial command interpreter. To invoke `login` in this fashion, type:

```
exec login
```

from the initial shell. The C shell and Korn shell have their own builtins of `login`. See `ksh(1)` and `csh(1)` for descriptions of `login` builtins and usage.

`login` asks for your user name, if it is not supplied as an argument, and your password, if appropriate. Where possible, echoing is turned off while you type your password, so it will not appear on the written record of the session.

If you make any mistake in the login procedure, the message:

```
Login incorrect
```

is printed and a new login prompt will appear. If you make five incorrect login attempts, all five may be logged in `/var/adm/loginlog`, if it exists. The TTY line will be dropped.

If password aging is turned on and the password has "aged" (see `passwd(1)` for more information), the user is forced to change the password. In this case the `/etc/nsswitch.conf` file is consulted to determine password repositories (see `nsswitch.conf(4)`). The password update configurations supported are limited to the following five cases:

- `passwd: files`
- `passwd: files nis`
- `passwd: files nisplus`
- `passwd: compat (==> files nis)`
- `passwd: compat (==> files nisplus)`
  ```
  passwd_compat: nisplus
  ```

Failure to comply with the configurations will prevent the user from logging onto the system because `passwd(1)` will fail. If you do not complete the login successfully within a certain period of time, it is likely that you will be silently disconnected.

After a successful login, accounting files are updated. Device owner, group, and permissions are set according to the contents of the `/etc/logindevperm` file, and the time you last logged in is printed (see `logindevperm(4)`).
The user-ID, group-ID, supplementary group list, and working directory are initialized, and the command interpreter (usually ksh) is started.

The basic environment is initialized to:

\begin{verbatim}
HOME=your-login-directory
LOGNAME=your-login-name
PATH=/usr/bin:
SHELL=last-field-of-pw-entry
MAIL=/var/mail/
TZ=timezone-specification
\end{verbatim}

For Bourne shell and Korn shell logins, the shell executes /etc/profile and $HOME/.profile, if it exists. For C shell logins, the shell executes /etc/.login, $HOME/.cshrc, and $HOME/.login. The default /etc/profile and /etc/.profile files check quotas (see quota(1M)), print /etc/motd, and check for mail. None of the messages are printed if the file $HOME/.hushlogin exists. The name of the command interpreter is set to \texttt{- (dash)}, followed by the last component of the interpreter's path name, for example, \texttt{-sh}.

If the login-shell field in the password file (see passwd(4)) is empty, then the default command interpreter, /usr/bin/sh, is used. If this field is * (asterisk), then the named directory becomes the root directory. At that point, \texttt{login} is re-executed at the new level, which must have its own root structure.

The environment may be expanded or modified by supplying additional arguments to \texttt{login}, either at execution time or when \texttt{login} requests your login name. The arguments may take either the form \texttt{xxx} or \texttt{xxx=yyy}. Arguments without an \texttt{=} (equal sign) are placed in the environment as:

\begin{verbatim}
Ln=xxx
\end{verbatim}

where \texttt{n} is a number starting at 0 and is incremented each time a new variable name is required. Variables containing an \texttt{=} (equal sign) are placed in the environment without modification. If they already appear in the environment, then they replace the older values.

There are two exceptions: The variables \texttt{PATH} and \texttt{SHELL} cannot be changed. This prevents people logged into restricted shell environments from spawning secondary shells that are not restricted. \texttt{login} understands simple single-character quoting conventions. Typing a \texttt{\} (backslash) in front of a character quotes it and allows the inclusion of such characters as spaces and tabs.

Alternatively, you can pass the current environment by supplying the \texttt{-p} flag to \texttt{login}. This flag indicates that all currently defined environment variables should be passed, if possible, to the new environment. This option does not bypass any environment variable restrictions mentioned above. Environment variables specified on the login line take precedence, if a variable is passed by both methods.
To enable remote logins by root, edit the `/etc/default/login` file by inserting a `#` (pound sign) before the `CONSOLE=/dev/console` entry. See FILES.

### SECURITY

The `login` command uses `pam(3PAM)` for authentication, account management, session management, and password management. The PAM configuration policy, listed through `/etc/pam.conf`, specifies the modules to be used for `login`. Here is a partial `pam.conf` file with entries for the `login` command using the UNIX authentication, account management, and session management modules:

```
login auth required pam_authtok_get.so.1
login auth required pam_dhkeys.so.1
login auth required pam_unix_auth.so.1
login auth required pam_dial_auth.so.1

login account requisite pam_roles.so.1
login account required pam_projects.so.1
login account required pam_unix_account.so.1

login session required pam_unix_session.so.1
```

The Password Management stack looks like the following:

```
other password required pam_dhkeys.so.1
other password requisite pam_authtok_get.so.1
other password requisite pam_authtok_check.so.1
other password required pam_authtok_store.so.1
```

If there are no entries for the service, then the entries for the "other" service will be used. If multiple authentication modules are listed, then the user may be prompted for multiple passwords.

When `login` is invoked through `rlogin` or `telnetd`, the service name used by PAM is `rlogin` or `telnet`, respectively.

### OPTIONS

The following options are supported:

- `-d device`
  - `login` accepts a device option, `device`. `device` is taken to be the path name of the TTY port `login` is to operate on. The use of the device option can be expected to improve `login` performance, since `login` will not need to call `ttyname(3C)`. The `-d` option is available only to users whose UID and effective UID are root. Any other attempt to use `-d` will cause `login` to quietly exit.

- `-h hostname [ terminal ]`
  - Used by `in.telnetd(1M)` to pass information about the remote host and terminal type.

- `-p`
  - Used to pass environment variables to the `login` shell.
login(1)

- `hostname` Used by in.rlogind(1M) to pass information about the remote host.

**EXIT STATUS**

The following exit values are returned:

- 0  Successful operation.
- non-zero  Error.

**FILES**

- `$HOME/.cshrc` initial commands for each csh
- `$HOME/.hushlogin` suppresses login messages
- `$HOME/.login` user’s login commands for csh
- `$HOME/.profile` user’s login commands for sh and ksh
- `$HOME/.rhosts` private list of trusted hostname/username combinations
- `/etc/.login` system-wide csh login commands
- `/etc/issue` issue or project identification
- `/etc/logindenvperm` login-based device permissions
- `/etc/motd` message-of-the-day
- `/etc/nologin` message displayed to users attempting to login during machine shutdown
- `/etc/passwd` password file
- `/etc/profile` system-wide sh and ksh login commands
- `/etc/shadow` list of users’ encrypted passwords
- `/usr/bin/sh` user’s default command interpreter
- `/var/adm/lastlog` time of last login
- `/var/adm/loginlog` record of failed login attempts
- `/var/adm/utmpx` accounting
- `/var/adm/wtmpx` accounting
- `/var/mail/your-name` mailbox for user `your-name`
- `/etc/default/login` Default value can be set for the following flags in
  `/etc/default/login`. For example:
  `TIMEZONE=EST5EDT`

  **TIMEZONE**
  Sets the TZ environment variable of the shell (see `environ(5)`).

  **HZ**
  Sets the HZ environment variable of the shell.
ULIMIT
Sets the file size limit for the login. Units are disk blocks. Default is zero (no limit).

CONSOLE
If set, root can login on that device only. This will not prevent execution of remote commands with rsh(1). Comment out this line to allow login by root.

PASSREQ
Determines if login requires a non-null password.

ALTSHELL
Determines if login should set the SHELL environment variable.

PATH
Sets the initial shell PATH variable.

SUPATH
Sets the initial shell PATH variable for root.

TIMEOUT
Sets the number of seconds (between 0 and 900) to wait before abandoning a login session.

UMASK
Sets the initial shell file creation mode mask. See umask(1).

SYSLOG
Determines whether the syslog(3C) LOG_AUTH facility should be used to log all root logins at level LOG_NOTICE and multiple failed login attempts at LOG_CRIT.

DISABLETIME
If present, and greater than zero, the number of seconds that login will wait after RETRIES failed attempts or the PAM framework returns PAM_ABORT. Default is 20 seconds. Minimum is 0 seconds. No maximum is imposed.

SLEEPTIME
If present, sets the number of seconds to wait before the login failure message is printed to the screen. This is for any login failure other than PAM_ABORT. Another login attempt is allowed, providing RETRIES has not been reached or the PAM framework is returned PAM_MAXTRIES. Default is 4 seconds. Minimum is 0 seconds. Maximum is 5 seconds.
RETRIES
Sets the number of retries for logging in (see pam(3PAM)). The default is 5.

SYSLOG_FAILED_LOGINS
Used to determine how many failed login attempts will be allowed by the system before a failed login message is logged, using the syslog(3C) LOG_NOTICE facility. For example, if the variable is set to 0, login will log all failed login attempts.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO
csh(1), exit(1), ksh(1), mail(1), mailx(1), newgrp(1), passwd(1), rlogin(1), rsh(1), sh(1), shell_builtins(1), telnet(1), umask(1), in.rlogind(1M), in.telnetd(1M), logins(1M), quota(1M), su(1M), syslogd(1M), useradd(1M), userdel(1M), pam(3PAM), rcmd(3SOCKET), syslog(3C), ttyname(3C), auth_attr(4), exec_attr(4), hosts.equiv(4), issue(4), logindevperm(4), loginlog(4), nologin(4), nsswitch.conf(4), pam.conf(4), passwd(4), profile(4), shadow(4), user_attr(4), utmpx(4), wtmpx(4), attributes(5), environ(5), pam_unix_account(5), pam_unix_auth(5), pam_unix_session(5), pam_authtok_check(5), pam_authtok_get(5), pam_authtok_store(5), pam_dhkeys(5), pam_passwd_auth(5), termio(7I)

DIAGNOSTICS
Login incorrect
The user name or the password cannot be matched.

Not on system console
Root login denied. Check the CONSOLE setting in /etc/default/login.

No directory! Logging in with home=/
The user’s home directory named in the passwd(4) database cannot be found or has the wrong permissions. Contact your system administrator.

No shell
Cannot execute the shell named in the passwd(4) database. Contact your system administrator.

NO LOGINS: System going down in N minutes
The machine is in the process of being shut down and logins have been disabled.

WARNINGS
Users with a UID greater than 76695844 are not subject to password aging, and the system does not record their last login time.
If you use the CONSOLE setting to disable root logins, you should arrange that remote command execution by root is also disabled. See rsh(1), rcmd(3SOCKET), and hosts.equiv(4) for further details.

NOTES

The pam_unix(5) module might not be supported in a future release. Similar functionality is provided by pam_unix_account(5), pam_unix_auth(5), pam_unix_session(5), pam_authtok_check(5), pam_authtok_get(5), pam_authtok_store(5), pam_dhkeys(5), and pam_passwd_auth(5).
logname(1)

NAME | logname – return user’s login name

SYNOPSIS | logname

DESCRIPTION | The logname utility will write the user’s login name to standard output. The login name is the string that would be returned by the getlogin(3C) function. Under the conditions where getlogin() would fail, logname will write a diagnostic message to standard error and exit with a non-zero exit status.

ENVIRONMENT VARIABLES | See environ(5) for descriptions of the following environment variables that affect the execution of logname: LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS | The following error values are returned:
- 0: Successful completion.
- >0: An error occurred.

FILES | /etc/profile: environment for user at login time
        /var/adm/utmpx: user and accounting information

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
</tbody>
</table>

SEE ALSO | env(1), login(1), getlogin(3C), utmpx(4), attributes(5), environ(5)
logout(1)

NAME
logout – shell built-in function to exit from a login session

csh logout

csh Terminate a login shell.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO csh(1), login(1), attributes(5)
NAME
look – find words in the system dictionary or lines in a sorted list

SYNOPSIS
/usr/bin/look [-d] [-f] [-tc] string [filename]

DESCRIPTION
The look command consults a sorted filename and prints all lines that begin with string.

If no filename is specified, look uses /usr/share/lib/dict/words with collating sequence -df.

look limits the length of a word to search for to 256 characters.

OPTIONS
-d Dictionary order. Only letters, digits, TAB and SPACE characters are used in comparisons.
-f Fold case. Upper case letters are not distinguished from lower case in comparisons.
-tc Set termination character. All characters to the right of c in string are ignored.

FILES
/usr/share/lib/dict/words spelling list

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
</tbody>
</table>

SEE ALSO
grep(1), sort(1), attributes(5)
**NAME**  
lookbib – find references in a bibliographic database

**SYNOPSIS**  
lookbib database

**DESCRIPTION**  
A bibliographic reference is a set of lines, constituting fields of bibliographic information. Each field starts on a line beginning with a ‘%’, followed by a key-letter, then a blank, and finally the contents of the field, which may continue until the next line starting with ‘%’.

The lookbib utility uses an inverted index made by indxbib to find sets of bibliographic references. It reads keywords typed after the ‘>’ prompt on the terminal, and retrieves records containing all these keywords. If nothing matches, nothing is returned except another ‘>’ prompt.

It is possible to search multiple databases, as long as they have a common index made by indxbib(1). In that case, only the first argument given to indxbib is specified to lookbib.

If lookbib does not find the index files (the .i[abc] files), it looks for a reference file with the same name as the argument, without the suffixes. It creates a file with a .ig suffix, suitable for use with fgrep (see grep(1)). lookbib then uses this fgrep file to find references. This method is simpler to use, but the .ig file is slower to use than the .i[abc] files, and does not allow the use of multiple reference files.

**FILES**  
x.ia
x.ib
x.ic  
index files
x.ig  
reference file

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWdoc</td>
</tr>
</tbody>
</table>

**SEE ALSO**  
addbib(1), grep(1), indxbib(1), refer(1), roffbib(1), sortbib(1), attributes(5)

**BUGS**  
Probably all dates should be indexed, since many disciplines refer to literature written in the 1800s or earlier.
lorder(1)

NAME  lorder – find ordering relation for an object or library archive

SYNOPSIS  lorder filename...

DESCRIPTION  The input is one or more object or library archive filenames (see ar(1)). The standard output is a list of pairs of object file or archive member names; the first file of the pair refers to external identifiers defined in the second. The output may be processed by tsort(1) to find an ordering of a library suitable for one-pass access by ld. Note that the link editor ld is capable of multiple passes over an archive in the portable archive format (see ar(3HEAD)) and does not require that lorder be used when building an archive. The usage of the lorder command may, however, allow for a more efficient access of the archive during the link edit process.

The following example builds a new library from existing .o files.

ar -cr library ' lorder *.o | tsort ' 

FILES  TMPDIR/*symref temporary files
       TMPDIR/*symdef temporary files
       TMPDIR usually /var/tmp but can be redefined by setting the environment variable TMPDIR (see tmpnam() in tmpnam(3C))

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWbtool</td>
</tr>
</tbody>
</table>

SEE ALSO  ar(1), ld(1), tsort(1), tmpnam(3C), ar(3HEAD), attributes(5)

NOTES  lorder will accept as input any object or archive file, regardless of its suffix, provided there is more than one input file. If there is but a single input file, its suffix must be .o.

The length of the filename for TMPDIR is limited to whatever sed allows.
NAME
lp – submit print request

SYNOPSIS
   [-H special-handling] [-n number] [-o option] [-P page-list]
   [-q priority-level] [-S character-set | print-wheel] [-t title] [-T content-type
   [-r]] [-y mode-list] [file...]

   [-H special-handling] [-n number] [-o option] [-P page-list]
   [-q priority-level] [-S character-set | print-wheel] [-t title] [-T content-type
   [-r]] [-y mode-list]

DESCRIPTION
The lp utility submits print requests to a destination. There are two formats of the lp
command.

The first form of lp prints files (file) and associated information (collectively called a
print request). If file is not specified, lp assumes the standard input. Use a hyphen (−) with file to specify the standard input. Files are printed in the order in which they
appear on the command line.

The second form of lp changes print request options. This form of lp can only be
used on a Solaris 2.6 Operating Environment or compatible versions of the LP print
server. The print request identified by request-ID is changed according to the printing
options specified. The printing options available are the same as those with the first
form of the lp. If the request has finished printing when the lp command is executed,
the change is rejected. If the request is in the process of printing, it will be stopped and
restarted from the beginning (unless the -P option has been given).

The print client commands locate destination information using the “printers”
database in the name service switch. See nsswitch.conf(4), printers(4), and
printers.conf(4) for details.

OPTIONS
Printers that have a 4.x or BSD-based print server are not configured to handle BSD
protocol extensions. lp handles print requests sent to such destinations differently (see
NOTES).

The following options are supported:

-c
   Copies file before printing.

   Unless -c is specified, users should not remove any file
   before the print request has completely printed.
   Changes made to file after the print request is made but
   before it is printed will be reflected in the printed
   output. file will be linked (as opposed to copied).

-d destination
   Prints file on a specific destination. The -d option is
   used to set the destination only when the job is first
   created. (Note: To move existing jobs to a different
   destination, see lpmove(1M).) destination can be either
   a printer or a class of printers (see lpadmin(1M)).
Specify destination using atomic, POSIX-style
(server:destination), or Federated Naming Service (FNS)
(.../service/printer/...) names. See
printers.conf(4) for information regarding the
naming conventions for atomic and FNS names and
standards(5) for information regarding POSIX.

- `f` form-name
  Prints file on form-name. The LP print service ensures
  that the form is mounted on the printer. The print
  request is rejected if the printer does not support
  form-name, if form-name is not defined for the system, or
  if the user is not allowed to use form-name (see
  lpforms(1M)).

- `H` special-handling
  Prints the print request according to the value of
  special-handling. The following special-handling values
  are acceptable:

  **hold**
  Do not print the print request until notified. If
  printing has already begun, stop it. Other print
  requests will go ahead of a request that has been put
  on hold (held print request) until the print request is
  resumed.

  **resume**
  Resume a held print request. If the print request had
  begun to print when held, it will be the next print
  request printed, unless it is superseded by an
  immediate print request.

  **immediate**
  Print the print request next. If more than one print
  request is assigned, the most recent print request is
  printed next. If a print request is currently printing
  on the desired printer, a hold request must be
  issued to allow the immediate request to print. The
  immediate request is only available to LP
  administrators.

- `i` request-ID
  Changes options for the print request identified by
  request-ID. There must be a space between `-i` and
  request-ID. This option applies only to jobs that are in a
  local queue on a print server.

- `m`
  Sends mail after file has printed (see mail(1)). By
  default, no mail is sent upon normal completion of a
  print request.

- `n` number
  Prints a specific number of copies of file. Specify number
  as a digit. The default for number is 1.
Specifies printer-dependent options. Specify several options by specifying `-o option` multiple times (`-o option -o option -o option`). Printer-dependent options may also be specified using the `-o` keyletter once, followed by a list of options enclosed in double quotes (`-o "option option option"`). The following options are valid:

**nobanner**

Does not print a banner page with the request. This option can be disallowed by the LP administrator.

**nofilebreak**

Prints multiple files without inserting a form feed between them.

**length=numberi | numberc | number**

Prints the print request with pages of a specific length in inches, centimeters, or number of lines. Append the letter `i` for inches or `c` for centimeters to `number`. Indicate the number of lines by specifying `number` alone. `length=66` indicates a page length of 66 lines. `length=11i` indicates a page length of 11 inches. `length=27.94c` indicates a page length of 27.94 centimeters.

This option may not be used with the `-f` option.

**width=numberi | numberc | number**

Prints the print request with pages of a specific width in inches, centimeters, or number of columns. Append the letter `i` for inches or `c` for centimeters to `number`. Indicate the number of columns by specifying `number` alone. `width=65` indicates a page width of 65 columns. `width=6.5i` indicates a page width of 6.5 inches. `width=10c` indicates a page width of 10 centimeters.

This option may not be used with the `-f` option.

**lpi=number**

Prints the print request with the line pitch set to `number` lines in an inch. Use `number` to specify the number of lines in an inch.

This option may not be used with the `-f` option.

**cpi=n|pica|elite|compressed**

Prints the print request with the character pitch set to `number` characters in an inch. Use `number` to specify the number of characters in an inch. Use
pica to set character pitch to pica (10 characters per inch), or elite to set character pitch to elite (12 characters per inch) Use compressed to set character pitch to as many characters as the printer can handle. There is no standard number of characters per inch for all printers; see the terminfo database (see terminfo(4)) for the default character pitch for your printer. This option may not be used with the -E option.

\texttt{stty=stty-option-list}

Prints the request using a list of options valid for the \texttt{stty} command (see \texttt{stty(1)}). Enclose the list in single quotes (‘’) if it contains blanks.

\texttt{-P page-list}

Prints the pages specified in \texttt{page-list} in ascending order. Specify \texttt{page-list} as a of range of numbers, single page number, or a combination of both.

The \texttt{-P} option can only be used if there is a filter available to handle it; otherwise, the print request will be rejected.

\texttt{-p}

Enables notification on completion of the print request. Delivery of the notification is dependent on additional software.

\texttt{-q priority-level}

Assigns the print request a priority in the print queue. Specify \texttt{priority-level} as an integer between from 0 and 39. Use 0 to indicate the highest priority; 39 to indicate the lowest priority. If no priority is specified, the default priority for a print service is assigned by the LP administrator. The LP administrator may also assign a default priority to individual users.

\texttt{-s}

Suppresses the display of messages sent from \texttt{lp}.

\texttt{-S character-set | -S print-wheel}

Prints the request using the \texttt{character-set} or \texttt{print-wheel}. If a form was requested and requires a character set or print wheel other than the one specified with the \texttt{-S} option, the request is rejected. Printers using mountable print wheels or font cartridges use the print wheel or font cartridge mounted at the time of the print request, unless the \texttt{-S} option is specified.

Printers Using Print Wheels: If \texttt{print wheel} is not one listed by the LP administrator as acceptable for the printer the request is rejected unless the print wheel is already mounted on the printer.
Printers Using Selectable or Programmable Character Sets: If the `-S` option is not specified, `lp` uses the standard character set. If `character-set` is not defined in the `terminfo` database for the printer (see `terminfo(4)`), or is not an alias defined by the LP administrator, the request is rejected.

```
-t title
```

Prints a title on the banner page of the output. Enclose `title` in quotes if it contains blanks. If `title` is not not specified, the name of the file is printed on the banner page.

```
-T content-type [-r]
```

Prints the request on a printer that can support the specified `content-type`. If no printer accepts this type directly, a filter will be used to convert the content into an acceptable type. If the `-r` option is specified, a filter will not be used. If `-r` is specified, and no printer accepts the `content-type` directly, the request is rejected. If the `content-type` is not acceptable to any printer, either directly or with a filter, the request is rejected.

```
-w
```

Writes a message on the user's terminal after the files have been printed. If the user is not logged in, then mail will be sent instead.

```
-y mode-list
```

Prints the request according to the printing modes listed in `mode-list`. The allowed values for `mode-list` are locally defined.

This option may be used only if there is a filter available to handle it; otherwise, the print request will be rejected.

**OPERANDS**
The following operand is supported:

```
file
```

The name of the file to be printed. Specify `file` as a pathname or as a hyphen (`-`) to indicate the standard input. If `file` is not specified, `lp` uses the standard input.

**USAGE**
See `largefile(5)` for the description of the behavior of `lp` when encountering files greater than or equal to 2 Gbyte (\(2^{31}\) bytes).

**EXIT STATUS**
The following exit values are returned:

```
0      Successful completion.
non-zero      An error occurred.
```

**FILES**

```
/var/spool/lp/*     LP print queue.
$HOME/.printers      User-configurable printer database.
/etc/printers.conf   System printer configuration database.
```
printers.confbyname NIS version of /etc/printers.conf.
printers.org_dir NIS+ version of /etc/printers.conf.
fns.ctx_dir.domain FNS version of /etc/printers.conf.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWpcu</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled (see NOTES)</td>
</tr>
</tbody>
</table>

SEE ALSO
cancel(1), enable(1), lpq(1B), lpr(1B), lprm(1B), lpstat(1), mail(1),
postprint(1), pr(1), stty(1), accept(1M), lpadm(1M), lpfilter(1M),
lpforms(1M), lpmove(1M), lpsched(1M), lpshut(1M), lpsystem(1M),
lpusers(1M), nsswitch.conf(4), printers(4), printers.conf(4), terminfo(4),
attributes(5), environ(5), largefile(5), standards(5)

NOTES
CSI-capability assumes that printer names are composed of ASCII characters.

Print jobs are assumed to contain one type of data. That type of data is either specified
on the command line or autodetected (simple, PostScript) based on the contents of the
first file in the job.

Printers that have a 4.x or BSD-based print server, are not configured to handle BSD
protocol extensions. lp handles print requests sent to such printers in the following
ways:

1. Print requests with more than 52 filenames will be truncated to 52 files. lp displays
   a warning message.
2. The -f, -H, -o, -P, -p, -q, -s, -T, and -y options may require a protocol
   extension to pass to a print server. If lp cannot handle the print request, it displays
   a warning message.

   LP administrators enable protocol extensions by setting a printer’s bsdaddr entry
   in /etc/printers.conf. Changing the bsdaddr entry in
   /etc/printers.conf to:

   destination:bsdaddr=server, destination, Solaris
   generates a set of BSD print
   protocol extensions that can be processed by a Solaris print server. lp supports
   only Solaris protocol extensions at this time.
NAME

lpc – line printer control program

SYNOPSIS

/usr/ucb/lpc  [command  [parameter...]]

DESCRIPTION

The lpc utility controls the operation of local printers.

Use lpc to perform the following functions:

- start or stop a printer,
- disable or enable a printer’s spooling queue,
- rearrange the order of jobs in a print queue, or
- display the status of a printer print queue and printer daemon.

lpc can be run from the command line or interactively. Specifying lpc with the optional command and parameter arguments causes lpc to interpret the first argument as an lpc command, and all other arguments as parameters to that command. Specifying lpc without arguments causes it to run interactively, prompting the user for lpc commands with lpc>. By redirecting the standard input, lpc can read commands from a file.

USAGE

lpc commands may be typed in their entirety or abbreviated to an unambiguous substring. Some lpc commands are available to all users; others are available only to super-users.

All users may execute the following lpc commands:

? [command ...] | help [command ...]
  Displays a short description of command. command is an lpc command. If command
  is not specified, displays a list of lpc commands.

  exit | quit
  Exits from lpc.

  status [all | printer ...]
  Displays the status of print daemons and print queues. all specifies that this
  command is performed on all locally attached printers. printer indicates this
  command is performed on specific printers. Specify printer as an atomic name. See
  printers.conf(4) for information regarding naming conventions for atomic
  names.

Only a super-user may execute the following lpc commands:

  abort [all | printer ...]
  Terminates an active spooling daemon. Disables printing (by preventing new
demons from being started by lpr(1B)) for printer. all specifies this command is
performed on all locally attached printers. printer indicates this command is
performed on specific printers. Specify printer as an atomic name. See
printers.conf(4) for information regarding naming conventions for atomic
names.
clean [all | printer ...]
Removes files created in the print spool directory by the print daemon from printer’s print queue. all specifies that this command is performed on all locally attached printers, printer indicates this command is performed on specific printers. Specify printer as an atomic name. See printers.conf(4) for information regarding naming conventions for atomic names.

disable [all | printer ...]
Turns off the print queue for printer. Prevents new printer jobs from being entered into the print queue for printer by lpr(1B). all specifies that this command is performed on all locally attached printers. printer indicates this command is performed on specific printers. Specify printer as an atomic name. See printers.conf(4) for information regarding naming conventions for atomic names.

down [all | printer ...] [message]
Turns the queue for printer off and disables printing on printer. Inserts message in the printer status file. message does not need to be quoted; multiple arguments to message are treated as arguments are to echo(1). Use down to take a printer down and inform users. all specifies that this command is performed on all locally attached printers. printer indicates this command is performed on specific printers. Specify printer as an atomic name. See printers.conf(4) for information regarding naming conventions for atomic names.

enable [all | printer ...]
Enables lpr(1B) to add new jobs in the spool queue. all specifies that this command is performed on all locally attached printers. printer indicates this command is performed on specific printers. Specify printer as an atomic name. See printers.conf(4) for information regarding naming conventions for atomic names.

restart [all | printer ...]
Attempts to start a new printer daemon. restart is useful when a print daemon dies unexpectedly and leaves jobs in the print queue. all specifies that this command is performed on all locally attached printers. printer indicates this command is performed on specific printers. Specify printer as an atomic name. See printers.conf(4) for information regarding naming conventions for atomic names.

start [all | printer ...]
Enables printing. Starts a spooling daemon for the printer. all specifies that this command is performed on all locally attached printers. printer indicates the command is performed on specific printers. Specify printer as an atomic name. See printers.conf(4) for information regarding naming conventions for atomic names.

stop [all | printer ...]
Stops a spooling daemon after the current job is complete. Disables printing at that time. all specifies that this command is performed on all locally attached printers.
printer indicates this command is performed on specific printers. Specify printer as an atomic name. See printers.conf(4) for information regarding naming conventions for atomic names.

topq printer [request-ID ...] [user ...]
Moves request-ID or print jobs belonging to user on printer to the beginning of the print queue. Specify user as a user's login name. Specify printer as an atomic name. See printers.conf(4) for information regarding naming conventions for atomic names.

up [all | printer ...]
Turns the queue for printer on and enables printing on printer. Deletes the message in the printer status file (inserted by down). Use up to undo the effects of down. all specifies that this command is performed on all locally attached printers. printer indicates this command is performed on specific printers. Specify printer as an atomic name. See printers.conf(4) for information regarding naming conventions for atomic names.

EXIT STATUS
The following exit values are returned:
0 Successful completion.
non-zero An error occurred.

FILES
/var/spool/lp/* LP print queue.
/var/spool/lp/system/pstatus Printer status information file.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscplp</td>
</tr>
</tbody>
</table>

SEE ALSO
echo(1), lpq(1B), lpr(1B), lprm(1B), lpstat(1), lpsched(1M), lpshut(1M), printers.conf(4), attributes(5)

DIAGNOSTICS
Ambiguous command
Indicates that the lpc command or abbreviation matches more than one command.

?Invalid command
Indicates that the lpc command or abbreviation is not recognized.

?Privileged command
Indicates that the lpc command or abbreviation can be executed only by a super-user.

lpc: printer : unknown printer to the print service
Indicates that printer does not exist in the LP database. Check that printer was correctly specified. Use lpstat -p or the status command (see lpstat(1) or USAGE) to check the status of printers.
lpc: error on opening queue to spooler
   Indicates that the connection to lpsched failed. Usually means that the printer
   server has died or is hung. Use /usr/lib/lp/lpsched to check if the printer
   spooler daemon is running.

lpc: Can’t send message to LP print service

lpc: Can’t receive message from LP print service
   Indicates that the LP print service stopped. Contact the LP administrator.

lpc: Received unexpected message from LP print service
   Indicates a problem with the software. Contact the LP administrator.
The `lpq` utility displays the information about the contents of a print queue. A print queue is comprised of print requests that are waiting in the process of being printed.

`lpq` displays the following information to the standard output:
- the username of the person associated with a print request,
- the position of a print request in the print queue,
- the name of file or files comprising a print request,
- the job number of a print request, and
- the size of the file requested by a print request. File size is reported in bytes.

Normally, only as much information as will fit on one line is displayed. If the name of the input file associated with a print request is not available, the input file field indicates the standard input.

The print client commands locate destination information using the “printers” database in the name service switch. See `nsswitch.conf(4)`, `printers(4)`, and `printers.conf(4)` for details.

The following options are supported:
- `-P destination` Displays information about printer or class of printers (see `lpadmin(1M)`). Specify `destination` using atomic, POSIX-style (`server:destination`), or Federated Naming Service (FNS) (`.../service/printer/...`) names. See `printers.conf(4)` for information regarding the naming conventions for atomic and FNS names, and `standards(5)` for information regarding POSIX.
- `-l` Displays information in long format. Long format includes the name of the host from which a print request originated in the display.
- `+ [interval]` Displays information at specific time intervals. Stops displaying information when the print queue is empty. Clears the screen before reporting displaying the print queue. Specify `interval` as the number of seconds between displays. If `interval` is not specified only executes once.

The following operands are supported:
- `request-ID` The job number associated with a print request.
- `user` The name of the user about whose jobs `lpq` reports information. Specify `user` as a valid username.

The following exit values are returned:
0 Successful completion.
non-zero An error occurred.

FILES /var/spool/print/[cd]*
   Spooling directory and request files for jobs awaiting transfer.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscplp</td>
</tr>
</tbody>
</table>

SEE ALSO lp(1), lpc(1B), lpr(1B), lprm(1B), lpstat(1), lpadmin(1M), nsswitch.conf(4), printers(4), printers.conf(4), attributes(5), standards(5)
The `lpr` utility submits print requests to a destination. `lpr` prints files and associated information, collectively called a **print request**. If file is not specified, `lpr` assumes the standard input.

Print requests with more than 52 files specified will be truncated to 52 files. `lpr` displays a warning message.

The following options are supported:

- **-P destination** Prints file on a specific printer or class of printers (see `lpadmin(1M)`). Specify `destination` using atomic, POSIX-style (`server:destination`), or Federated Naming Service (FNS) (`../service/printer/...`) names. See `printers.conf(4)` for information regarding the naming conventions for atomic and FNS names, and `standards(5)` for information regarding POSIX.

- **-# number** Prints a specific number of copies. Specify `number` as a positive integer. The default for `number` is 1.

- **-C class** Prints `class` as the job classification on the banner page of the output. Enclose `class` in double quotes if it contains blanks. If `class` is not specified, the name of the system (as returned by `hostname`) is printed as the job classification. See `hostname(1)`.

- **-J job** Prints `job` as the job name on the banner page of the output. Enclose `job` in double quotes if it contains blanks. If `job` is not specified, file (or in the case of multiple files, the first file specified on the command line) is printed as the job name on the banner page of the output.

- **-T title** Prints a title on the page header of the output. Enclose `title` in double quotes if it contains blanks. The `-T` option is ignored unless it is specified with the `-p` filter option.

- **-i indent** Indents the output a specific number of SPACE characters. Use `indent` to indicate the number of SPACE characters to be indented. Specify `indent` as a positive integer.
integer. If the optional argument to indent is not specified, then eight SPACE characters is the default. The -i option is ignored unless it is specified with the -p filter option.

-1-2-3-4 font
Mounts the specified font in the font position 1, 2, 3, or 4. Specify font as a valid font name.

-w cols
Prints file with pages of a specific width. cols indicates the number of columns wide. The -w option is ignored unless it is specified with the -p filter option.

-m
Sends mail after file has printed. See mail(1). By default, no mail is sent upon normal completion of a print request.

-h
Suppresses printing of the banner page of the output.

-s
Uses full pathnames (as opposed to symbolic links) to file rather than trying to copy them. This means file should not be modified or removed until it has completed printing. Option -s only prevents copies of local files from being made on the local machine. Option -s only works with specified files. If the lpr command is at the end of a pipeline, file is copied to the spool.

-filter_option
Notifies the print spooler that file is not a standard text file. Enables the spooling daemon to use the appropriate filters to print file.

filter_options offer a standard user interface. All filter options may not be available for, or applicable to, all printers.

Specify filter_option as a single character.

If filter_option is not specified and the printer can interpret PostScript®, inserting ‘%!’ as the first two characters of file causes file to be interpreted as PostScript.

The following filter options are supported:

p Use pr to format the files. See pr(1).

l Print control characters and suppress page breaks.

t file contains troff (cat phototypesetter) binary data.
lpr(1B)

n  file contains ditroff data from device independent troff.

d  file contains tex data in DVI format from Stanford.

g  file contains standard plot data produced by plot(1B) routines.

v  file contains a raster image. printer must support an appropriate imaging model such as PostScript in order to print the image.

c  file contains data produced by cifplot.

f  Interprets the first character of each line as a standard FORTRAN carriage control character.

OPERANDS  The following operand is supported:

file  The name of the file to be printed. Specify file as a pathname. If file is not specified, lpr uses the standard input.

USAGE  See largefile(5) for the description of the behavior of lpr when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXIT STATUS  The following exit values are returned:

0  Successful completion.

non-zero  An error occurred.

FILES  
/var/spool/print/.seq  File containing the sequence numbers for job ID assignment.

/var/spool/print/[cd]f*  Spooling directories and files.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscplp</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled (see NOTES)</td>
</tr>
</tbody>
</table>

SEE ALSO  hostname(1), lp(1), lpc(1B), lpq(1B), lprm(1B), lpstat(1), mail(1), plot(1B), pr(1), troff(1), ladmin(1M), nsswitch.conf(4), printers(4), printers.conf(4), attributes(5), largefile(5), standards(5)

DIAGNOSTICS  lpr: destination |: unknown destination

destination was not found in the LP configuration database. Usually this is a typing mistake; however, it may indicate that the destination does not exist on the system.
Use `lpstat -p` to display information about the status of the print service.

**NOTES**

`lpr` is CSI-enabled except for the *printer* name.

Print jobs are assumed to contain one type of data. That type of data is either specified on the command line or autodetected (simple, PostScript) based on the contents of the first file in the job.
NAME
lprm – remove print requests from the print queue

SYNOPSIS
/usr/ucb/lprm [-p destination] [-] [request-ID...] [user...]

DESCRIPTION
The lprm utility removes print requests (request-ID) from the print queue.

Without arguments, lprm deletes the current print request. lprm reports the name of
the file associated with print requests that it removes. lprm is silent if there are no
applicable print requests to remove.

Users can only remove print requests associated with their user name. See NOTES. If a
super-user executes lprm and specifies the user operand, lprm removes all print
requests belonging to the specified user.

The print client commands locate destination information using the “printers”
database in the name service switch. See nsswitch.conf(4), printers(4), and
printers.conf(4) for details.

OPTIONS
The following options are supported:

-p destination The name of the printer or class of printers (see lpadmin(1M))
from which to remove print requests. Specify destination using
atomic, POSIX-style (server:destination), or Federated Naming
Service (FNS) (.../service/printer/...) names. See
printers.conf(4) for information regarding the naming
conventions for atomic and FNS names, and standards(5) for
information regarding POSIX.

If a user specifies this option, removes all print requests owned by
that user. If a super-user specifies this option, removes all requests
in the print queue. Job ownership is determined by the user’s login
name and host name on the machine from which lpr was
executed. See NOTES.

OPERANDS
The following operands are supported.

user Removes print requests associated with a specific user. Specify user
as a valid user name. This option can only be used by a super-user.

request-ID Removes a specific print request. Specify request-ID as the job
number (Job) associated with a print request and reported by
lpq. See lpq(1B).

EXAMPLES
EXAMPLE 1 Removing a print request

The following example removes request-ID 385 from destination killtree:

example% lprm -P killtree 385

EXIT STATUS
The following exit values are returned:

0 Successful completion.
lprm(1B)

non-zero An error occurred.

FILES
/var/spool/print/[cd]* Spooling directories and files.

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

<table>
<thead>
<tr>
<th>ATTRIBUTES TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscplp</td>
</tr>
</tbody>
</table>

SEE ALSO lp(1), lpc(1B), lpq(1B), lpr(1B), lpstat(1), lpadmin(1M), nsswitch.conf(4), printers(4), printers.conf(4), attributes(5), standards(5)

NOTES Users can only remove print requests associated with their user name. By default, users can only remove print requests on the host from which the print request was submitted. If a super-user has set user-equivalence=true in /etc/printers.conf on the print server, users can remove print requests associated with their user name on any host. Super-users can remove print requests on the host from which the print request was submitted. Super-users can also remove print requests from the print server.
lpstat – print information about the status of the print service

SYNOPSIS

lpstat [-d] [-r] [-R] [-s] [-t] [-a [list]] [-c [list]] [-f [list] [-l]]
[-o [list]] [-p [list] [-D] [-l]] [-S [list] [-l]] [-u [login-ID]]
[-v [list]]

DESCRIPTION

The lpstat utility displays information about the current status of the LP print service to standard output.

If no options are given, lpstat prints the status of all the user’s print requests made by lp. (see lp(1)). Any arguments that are not options are assumed to be request-IDs as returned by lp. The lpstat command prints the status of such requests. options may appear in any order and may be repeated and intermixed with other arguments. Some key letters may be followed by an optional list that can be in one of two forms: a list of items separated from one another by a comma, or a list of items separated from one another by spaces enclosed in quotes. For example:

example% lpstat -u "user1 user2 user3"

Specifying all after any key letter that takes list as an argument causes all information relevant to the key letter to be printed. For example, the command:

example% lpstat -o all

prints the status of all output requests.

The omission of a list following such key letters causes all information relevant to the key letter to be printed. For example, the command:

example% lpstat -o

prints the status of all output requests.

The print client commands locate destination information using the “printers” database in the name service switch. See nsswitch.conf(4), printers(4), and printers.conf(4) for details.

OPTIONS

The following options are supported on all platforms.

- -d
  Prints the default destination for output requests.

- o [list]
  Prints the status of output requests. list is a list of intermixed printer names, class names, and request-IDs. The key letter -o may be omitted. Specify printer and class names using atomic, POSIX-style (server:destination), or Federated Naming Service (FNS) (.../service/printer/...) names. See printers.conf(4) for information regarding the naming conventions for atomic and FNS names, and standards(5) for information regarding POSIX.
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-r</code></td>
<td>Prints the status of the LP request scheduler.</td>
</tr>
<tr>
<td><code>-R</code></td>
<td>Prints a number showing the position of each request in the print queue.</td>
</tr>
<tr>
<td><code>-s</code></td>
<td>Prints a status summary, including the status of the LP scheduler, the default destination, a list of printers and their associated devices, a list of the machines sharing print services, a list of all forms currently mounted, and a list of all recognized character sets and print wheels.</td>
</tr>
<tr>
<td><code>-t</code></td>
<td>Prints all status information. This includes all the information obtained with the <code>-s</code> option, plus the acceptance and idle/busy status of all printers.</td>
</tr>
<tr>
<td><code>-u [login-ID-list]</code></td>
<td>Prints the status of output requests for users. The <code>login-ID-list</code> argument may include any or all of the following constructs:</td>
</tr>
<tr>
<td><code>login-ID</code></td>
<td>a user on any system</td>
</tr>
<tr>
<td><code>system_name:login-ID</code></td>
<td>a user on system <code>system_name</code></td>
</tr>
<tr>
<td><code>system_name:all</code></td>
<td>all users on system <code>system_name</code></td>
</tr>
<tr>
<td><code>all:login-ID</code></td>
<td>a user on all systems</td>
</tr>
<tr>
<td><code>all</code></td>
<td>all users on all systems</td>
</tr>
<tr>
<td><code>-v [list]</code></td>
<td>Prints the names of printers and the path names of the devices associated with them or remote system names for network printers. <code>list</code> is a list of printer names.</td>
</tr>
<tr>
<td><code>-a [list]</code></td>
<td>Reports whether print destinations are accepting requests. <code>list</code> is a list of intermixed printer names and class names.</td>
</tr>
<tr>
<td><code>-c [list]</code></td>
<td>Prints name of all classes and their members. <code>list</code> is a list of class names.</td>
</tr>
<tr>
<td><code>-f [list] [-l]</code></td>
<td>Prints a verification that the forms in <code>list</code> are recognized by the LP print service. <code>list</code> is a list of forms; the default is <code>all</code>. The <code>-l</code> option will list the form descriptions.</td>
</tr>
</tbody>
</table>
| `-p [list] [-D] [-l]` | Prints the status of printers. `list` is a list of printer names. If the `-D` option is given, a brief description is

The following options return accurate results only if they are issued from a Solaris 2.6 Operating Environment or compatible version of the LP print server.
lpstat(1)

printed for each printer in list. If the -1 option is given and the printer is on the local machine, a full description of each printer’s configuration is returned, including the form mounted, the acceptable content and printer types, a printer description, and the interface used.

-S [list] [-1]
Prints a verification that the character sets or the print wheels specified in list are recognized by the LP print service. Items in list can be character sets or print wheels; the default for the list is all. If the -1 option is given, each line is appended by a list of printers that can handle the print wheel or character set. The list also shows whether the print wheel or character set is mounted, or specifies the built-in character set into which it maps.

-d
Prints the default destination for output requests.

-o [list]
Prints the status of output requests. list is a list of intermixed printer names, class names, and request-IDs. The key letter -o may be omitted.

-r
Prints the status of the LP request scheduler.

-R
Prints a number showing the position of each request in the print queue.

-s
Prints a status summary, including the status of the LP scheduler, the default destination, a list of printers and their associated devices, a list of the machines sharing print services, a list of all forms currently mounted, and a list of all recognized character sets and print wheels.

-t
Prints all status information. This includes all the information obtained with the -s option, plus the acceptance and idle/busy status of all printers.

-u [login-ID-list]
Prints the status of output requests for users. The login-ID-list argument may include any or all of the following constructs:

login-ID
a user on any system

system_name!login-ID
a user on system system_name

system_name!all
all users on system system_name

all!login-ID
a user on all systems
lpstat(1)

all

all users on all systems

-v [list]  Prints the names of printers and the path names of the
devices associated with them or remote system names
for network printers. list is a list of printer names.

EXIT STATUS  The following exit values are returned:
0  Successful completion.
non-zero  An error occurred.

FILES  /var/spool/print/*  LP print queue.
$HOME/.printers  User-configurable printer database.
/etc/printers.conf  System configuration database.
printers.conf.bynmale  NIS version of /etc/printers.conf.
printers.org_dir  NIS+ version of /etc/printers.conf.
fns.ctx_dir.domain  FNS version of /etc/printers.conf.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWpcu</td>
</tr>
</tbody>
</table>

SEE ALSO  cancel(1), lp(1), lpq(1B), lpr(1B), lprm(1B), nsswitch.conf(4), printers(4),
printers.conf(4), attributes(5), standards(5)
NAME  lptest – generate line printer ripple pattern

SYNOPSIS  /usr/ucb/lptest  [length  [count]]

DESCRIPTION  The lptest utility writes the traditional ripple test pattern to the standard output. In 96 lines, the ripple test pattern prints all 96 printable ASCII characters in each position. The ripple test pattern was originally created to test printers. It is also useful for testing terminals, driving terminal ports, debugging, and performing tasks that require a quick supply of random data.

This command is obsolete.

OPTIONS  length  Specifies the length of the output line in characters. 79 characters is the default.

count  Specifies the number of output lines. 200 lines is the default. If count is specified, length must also be specified.

EXIT STATUS  The following exit values are returned:

0  Successful completion.

non-zero  An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpplp</td>
</tr>
</tbody>
</table>

SEE ALSO  attributes(5)
NAME | \texttt{ls} – list contents of directory  

SYNOPSIS | \texttt{/usr/bin/ls} [-aAbcCdfGhilMnopqrStux1@] \texttt{[file...]}  
\texttt{/usr/xpg4/bin/ls} [-aAbcCdfGhilMnopqrStux1@] \texttt{[file...]}  

DESCRIPTION | For each \textit{file} that is a directory, \texttt{ls} lists the contents of the directory. For each \textit{file} that is an ordinary file, \texttt{ls} repeats its name and any other information requested. The output is sorted alphabetically by default. When no argument is given, the current directory is listed. When several arguments are given, the arguments are first sorted appropriately, but file arguments appear before directories and their contents.

There are three major listing formats. The default format for output directed to a terminal is multi-column with entries sorted down the columns. The \texttt{-l} option allows single column output and \texttt{-m} enables stream output format. In order to determine output formats for the \texttt{-C}, \texttt{-x}, and \texttt{-m} options, \texttt{ls} uses an environment variable, \texttt{COLUMNS}, to determine the number of character positions available on one output line. If this variable is not set, the \texttt{terminfo(4)} database is used to determine the number of columns, based on the environment variable, \texttt{TERM}. If this information cannot be obtained, 80 columns are assumed.

The mode printed under the \texttt{-l} option consists of ten characters. The first character may be one of the following:

\begin{itemize}
  \item \texttt{d} The entry is a directory.
  \item \texttt{D} The entry is a door.
  \item \texttt{l} The entry is a symbolic link.
  \item \texttt{b} The entry is a block special file.
  \item \texttt{c} The entry is a character special file.
  \item \texttt{p} The entry is a FIFO (or “named pipe”) special file.
  \item \texttt{s} The entry is an \texttt{AF_UNIX} address family socket.
  \item \texttt{-} The entry is an ordinary file.
\end{itemize}

The next 9 characters are interpreted as three sets of three bits each. The first set refers to the owner’s permissions; the next to permissions of others in the user-group of the file; and the last to all others. Within each set, the three characters indicate permission to read, to write, and to execute the file as a program, respectively. For a directory, “execute” permission is interpreted to mean permission to search the directory for a specified file. The character after permissions is ACL indication. A plus sign is displayed if there is an ACL associated with the file. Nothing is displayed if there are just permissions.

\texttt{ls -l} (the \texttt{long list}) prints its output as follows for the POSIX locale:

\begin{verbatim}
-rwxrwxrwx+ 1 smith dev 10876 May 16 9:42 part2
\end{verbatim}
Reading from right to left, you see that the current directory holds one file, named `part2`. Next, the last time that file's contents were modified was 9:42 A.M. on May 16. The file contains 10,876 characters, or bytes. The owner of the file, or the user, belongs to the group `dev` (perhaps indicating “development”), and his or her login name is `smith`. The number, in this case 1, indicates the number of links to file `part2` (see `cp(1)`). The plus sign indicates that there is an ACL associated with the file. **Note:** If the `-@` option has been specified, the presence of extended attributes will supersede the presence of an ACL and the plus sign will be replaced with an ‘at’ sign (@). Finally, the dash and letters tell you that user, group, and others have permissions to read, write, and execute `part2`.

The execute (x) symbol here occupies the third position of the three-character sequence. A – in the third position would have indicated a denial of execution permissions.

The permissions are indicated as follows:

- **r** The file is readable.
- **w** The file is writable.
- **x** The file is executable.
- **−** The indicated permission is *not* granted.
- **s** The set-user-ID or set-group-ID bit is on, and the corresponding user or group execution bit is also on.
- **S** Undefined bit-state (the set-user-ID bit is on and the user execution bit is off).
- **t** The 1000 (octal) bit, or sticky bit, is on (see `chmod(1)`), and execution is on.
- **T** The 1000 bit is turned on, and execution is off (undefined bit-state).

<table>
<thead>
<tr>
<th>Directory Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/usr/bin/ls</code></td>
<td>Mandatory locking occurs during access (the set-group-ID bit is on and the group execution bit is off).</td>
</tr>
<tr>
<td><code>/usr/xpg4/bin/ls</code></td>
<td>Mandatory locking occurs during access (the set-group-ID bit is on and the group execution bit is off).</td>
</tr>
</tbody>
</table>

For user and group permissions, the third position is sometimes occupied by a character other than x or −. s also may occupy this position, referring to the state of the set-ID bit, whether it be the user’s or the group’s. The ability to assume the same ID as the user during execution is, for example, used during login when you begin as root but need to assume the identity of the user you login as.

In the case of the sequence of group permissions, 1 may occupy the third position. 1 refers to mandatory file and record locking. This permission describes a file’s ability to allow other files to lock its reading or writing permissions during access.

For others permissions, the third position may be occupied by t or T. These refer to the state of the sticky bit and execution permissions.
ls(1)

OPTIONS

The following options are supported:

- `-a` Lists all entries, including those that begin with a dot (.), which are
  normally not listed.

- `-A` Lists all entries, including those that begin with a dot (.), with the
  exception of the working directory (.) and the parent directory (..).

- `-b` Forces printing of non-printable characters to be in the octal \ddd notation.

- `-c` Uses time of last modification of the i-node (file created, mode changed,
  and so forth) for sorting (-t) or printing (-1 or -n).

- `-C` Multi-column output with entries sorted down the columns. This is the
  default output format.

- `-d` If an argument is a directory, lists only its name (not its contents). Often
  used with -1 to get the status of a directory.

- `-f` Forces each argument to be interpreted as a directory and list the name
  found in each slot. This option turns off -1, -t, -s, and -r, and turns on
  -a. The order is the order in which entries appear in the directory.

- `-F` Marks directories with a trailing slash (/), doors with a trailing
  greater-than sign (>), executable files with a trailing asterisk (*), FIFOs with
  a trailing vertical bar (|), symbolic links with a trailing 'at' sign (@), and
  AF_UNIX address family sockets with a trailing equals sign (=).

- `-g` The same as -1, except that the owner is not printed.

- `-h` All sizes are scaled to a human readable format, for example, 14K, 234M,
  2.7G, or 3.0T. Scaling is done by repetitively dividing by 1024.

- `-i` For each file, prints the i-node number in the first column of the report.

- `-l` Lists in long format, giving mode, ACL indication, number of links, owner,
  group, size in bytes, and time of last modification for each file (see above).
  If the file is a special file, the size field instead contains the major and
  minor device numbers. If the time of last modification is greater than six
  months ago, it is shown in the format ‘month date year’ for the POSIX
  locale. When the LC_TIME locale category is not set to the POSIX locale, a
  different format of the time field may be used. Files modified within six
  months show ‘month date time’. If the file is a symbolic link, the filename
  is printed followed by “→” and the path name of the referenced file.

- `-L` If an argument is a symbolic link, lists the file or directory the link
  references rather than the link itself.

- `-m` Streams output format. Files are listed across the page, separated by
  commas.

- `-n` The same as -1, except that the owner’s UID and group’s GID numbers are
  printed, rather than the associated character strings.

- `-o` The same as -1, except that the group is not printed.
-p  Puts a slash (/) after each filename if the file is a directory.
-q  Forces printing of non-printable characters in file names as the character question mark (?)
-r  Reverses the order of sort to get reverse alphabetic or oldest first as appropriate.
-R  Recursively lists subdirectories encountered.
-s  Gives size in blocks, including indirect blocks, for each entry.
-t  Sorts by time stamp (latest first) instead of by name. The default is the last modification time. (See -u and -c.)
-u  Uses time of last access instead of last modification for sorting (with the -t option) or printing (with the -l option).
-@ The same as -l, except that extended attribute information will supersede ACL information. An @ is displayed after the file permission bits for files that have extended attributes.
-x  Multi-column output with entries sorted across rather than down the page.
-1  Prints one entry per line of output.

/specifying more than one of the options in the following mutually exclusive pairs is not considered an error:
- C and -l (ell), - m and - l (ell), - x and - l (ell), - @ and - l (ell). The - l option overrides the other option specified in each pair.

/specifying more than one of the options in the following mutually exclusive pairs is not considered an error:
- C and - l (ell), - m and - l (ell), - x and - l (ell), - @ and - l (ell). The last option specified in each pair determines the output format.

/OPERANDS The following operand is supported:

/file A path name of a file to be written. If the file specified is not found, a diagnostic message will be output on standard error.

/USAGE See largefile(5) for the description of the behavior of ls when encountering files greater than or equal to 2 Gbyte (231 bytes).

/EXAMPLES EXAMPLE 1 File permissions

An example of a file’s permissions is:

- rwxr--r--

This describes a file that is readable, writable, and executable by the user and readable by the group and others.

Another example of a file’s permissions is:

- rwxr-xr-x
EXAMPLE 1 File permissions  (Continued)

This describes a file that is readable, writable, and executable by the user, readable and executable by the group and others, and allows its user-ID to be assumed, during execution, by the user presently executing it.

Another example of a file’s permissions is:

−rw−rw−−−

This describes a file that is readable and writable only by the user and the group and can be locked during access.

EXAMPLE 2 Printing the names of all files

This command prints the names of all files in the current directory, including those that begin with a dot (.), which normally do not print:

example% ls -a

EXAMPLE 3 Providing file information

Another example of a command line is:

example% ls -aisn

This command provides information on all files, including those that begin with a dot (a), the i-number—the memory address of the i-node associated with the file—printed in the left-hand column (i); the size (in blocks) of the files, printed in the column to the right of the i-numbers (s); finally, the report is displayed in the numeric version of the long list, printing the UID (instead of user name) and GID (instead of group name) numbers associated with the files.

When the sizes of the files in a directory are listed, a total count of blocks, including indirect blocks, is printed.

See environ(5) for descriptions of the following environment variables that affect the execution of ls: LC_COLLATE, LC_CTYPE, LC_TIME, LC_MESSAGES, NLSPATH, and TZ.

COLUMNS  Determines the user’s preferred column position width for writing multiple text-column output. If this variable contains a string representing a decimal integer, the ls utility calculates how many path name text columns to write (see -C) based on the width provided. If COLUMNS is not set or is invalid, 80 is used. The column width chosen to write the names of files in any given directory will be constant. File names will not be truncated to fit into the multiple text-column output.
EXIT STATUS
0    All information was written successfully.
>0   An error occurred.

FILES
/etc/group  group IDs for ls -l and ls -g
/etc/passwd user IDs for ls -l and ls -o
/usr/share/lib/terminfo/?.*/ terminal information database

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

<table>
<thead>
<tr>
<th>/usr/bin/ls</th>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
<td></td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
<td></td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Stable</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>/usr/xpg4/bin/ls</th>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWxcu4</td>
<td></td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
<td></td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Standard</td>
<td></td>
</tr>
</tbody>
</table>

SEE ALSO
chmod(1), cp(1), setfacl(1), terminfo(4), attributes(5), environ(5), fsattr(5), largefile(5), XPG4(5)

NOTES
Unprintable characters in file names may confuse the columnar output options.

The total block count will be incorrect if there are hard links among the files.

The sort order of ls output is affected by the locale and can be overridden by the LC_COLLATE environment variable. For example, if LC_COLLATE equals C, dot files appear first, followed by names beginning with upper-case letters, then followed by names beginning with lower-case letters. But if LC_COLLATE equals en_US.ISO8859-1, then leading dots as well as case are ignored in determining the sort order.
ls – list the contents of a directory

For each filename that is a directory, \texttt{ls} lists the contents of the directory; for each filename that is a file, \texttt{ls} repeats its name and any other information requested. By default, the output is sorted alphabetically. When no argument is given, the current directory is listed. When several arguments are given, the arguments are first sorted appropriately, but file arguments are processed before directories and their contents.

The mode printed under the \texttt{-l} option contains 10 characters interpreted as follows. If the first character is:

- \texttt{d} Entry is a directory.
- \texttt{D} Entry is a door.
- \texttt{b} Entry is a block-type special file.
- \texttt{c} Entry is a character-type special file.
- \texttt{l} Entry is a symbolic link.
- \texttt{p} Entry is a FIFO (also known as “named pipe”) special file.
- \texttt{s} Entry is an \texttt{AF\_UNIX} address family socket.
- \texttt{−} Entry is a plain file.

The next 9 characters are interpreted as three sets of three bits each. The first set refers to owner permissions; the next refers to permissions to others in the same user-group; and the last refers to all others. Within each set, the three characters indicate permission respectively to read, to write, or to execute the file as a program. For a directory, “execute” permission is interpreted to mean permission to search the directory. The permissions are indicated as follows:

- \texttt{r} The file is readable.
- \texttt{w} The file is writable.
- \texttt{x} The file is executable.
- \texttt{−} The indicated permission is not granted.

The group-execute permission character is given as \texttt{s} if the file has the set-group-id bit set; likewise the owner-execute permission character is given as \texttt{s} if the file has the set-user-id bit set.

The last character of the mode (normally \texttt{x} or \texttt{−}) is \texttt{true} if the 1000 bit of the mode is on. See \texttt{chmod(1)} for the meaning of this mode. The indications of set-ID and 1000 bits of the mode are capitalized (\texttt{S} and \texttt{T}, respectively) if the corresponding execute permission is not set.

A plus sign (\texttt{+}) appended to the list of permissions indicates that an ACL is associated with the file.
When the sizes of the files in a directory are listed, a total count of blocks, including indirect blocks, is printed.

**OPTIONS**

The following options are supported:

- **a** Lists all entries; in the absence of this option, entries whose names begin with a ’.’ are not listed (except for the privileged user, for whom `ls` normally prints even files that begin with a ’.’).

- **A** Same as -a, except that ’.’ and ‘. .’ are not listed.

- **c** Uses time of last edit (or last mode change) for sorting or printing.

- **C** Forces multi-column output, with entries sorted down the columns; for `ls`, this is the default when output is to a terminal.

- **d** If argument is a directory, lists only its name (not its contents); often used with -l to get the status of a directory.

- **f** Forces each argument to be interpreted as a directory and lists the name found in each slot. This option turns off -l, -t, -s, and -r, and turns on -a; the order is the order in which entries appear in the directory.

- **F** Marks directories with a trailing slash (/), doors with a trailing greater-than sign (>), executable files with a trailing asterisk (*), FIFOs with a trailing vertical bar (|), symbolic links with a trailing at-sign (@), and AF_UNIX address family sockets with a trailing equals sign (=).

- **g** For `ls`, shows the group ownership of the file in a long output.

- **i** For each file, prints the i-node number in the first column of the report.

- **l** Lists in long format, giving mode, ACL indication, number of links, owner, size in bytes, and time of last modification for each file. If the file is a special file the size field will instead contain the major and minor device numbers. If the time of last modification is greater than six months ago, it is shown in the format ’month date year’; files modified within six months show ’month date time’. If the file is a symbolic link, the pathname of the linked-to file is printed preceded by ’—>’.

- **L** If argument is a symbolic link, lists the file or directory the link references rather than the link itself.

- **q** Displays non-graphic characters in filenames as the character ?; for `ls`, this is the default when output is to a terminal.

- **r** Reverses the order of sort to get reverse alphabetic or oldest first as appropriate.

- **R** Recursively lists subdirectories encountered.

- **s** Gives size of each file, including any indirect blocks used to map the file, in kilobytes.

- **t** Sorts by time modified (latest first) instead of by name.
-u Uses time of last access instead of last modification for sorting (with the -t option) and/or printing (with the -l option).

-1 Forces one entry per line output format; this is the default when output is not to a terminal.

**OPERANDS**
The following operand is supported:

file A path name of a file to be listed. If the file specified is not found, a diagnostic message will be output on standard error.

**USAGE**
See largefile(5) for the description of the behavior of ls when encountering files greater than or equal to 2 Gbyte (231 bytes).

**FILES**
/etc/group to get group ID for 'ls -g'

/etc/passwd to get user IDs for 'ls -l' and 'ls -o'

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

**SEE ALSO**
ls(1), attributes(5), largefile(5)

**NOTES**
NEWLINE and TAB are considered printing characters in filenames.

The output device is assumed to be 80 columns wide.

The option setting based on whether the output is a teletype is undesirable as 'ls -s' is much different than 'ls -s | lpr'. On the other hand, not doing this setting would make old shell scripts which used ls almost certain losers.

Unprintable characters in file names may confuse the columnar output options.
NAME | m4 – macro processor 
DESCRIPTION | The m4 utility is a macro processor intended as a front end for C, assembler, and other languages. Each of the argument files is processed in order; if there are no files, or if a file is −, the standard input is read. The processed text is written on the standard output. 
Macro Syntax | Macro calls have the form: 
name (arg1, arg2, ..., argn) 
The ( must immediately follow the name of the macro. If the name of a defined macro is not followed by a (, it is deemed to be a call of that macro with no arguments. Potential macro names consist of alphanumeric characters and underscore (_), where the first character is not a digit. 
Leading unquoted blanks, TABs, and NEWLINEs are ignored while collecting arguments. Left and right single quotes are used to quote strings. The value of a quoted string is the string stripped of the quotes. 
Macro Processing | When a macro name is recognized, its arguments are collected by searching for a matching right parenthesis. If fewer arguments are supplied than are in the macro definition, the trailing arguments are taken to be NULL. Macro evaluation proceeds normally during the collection of the arguments, and any commas or right parentheses that happen to turn up within the value of a nested call are as effective as those in the original input text. After argument collection, the value of the macro is pushed back onto the input stream and rescanned. 
OPTIONS | The options and their effects are as follows: 
-e | Operate interactively. Interrupts are ignored and the output is unbuffered. 
-s | Enable line sync output for the C preprocessor (#line ...)
-bint | Change the size of the push-back and argument collection buffers from the default of 4,096.
-hint | Change the size of the symbol table hash array from the default of 199. The size should be prime.
-Sint | Change the size of the call stack from the default of 100 slots. Macros take three slots, and non-macro arguments take one.
-tint | Change the size of the token buffer from the default of 512 bytes.
To be effective, the above flags must appear before any file names and before any -D or -U flags:

- D name[=val]    Defines name to val or to NULL in val’s absence.
- U name           Undefines name.

OPERANDS

The following operand is supported:

file    A path name of a text file to be processed. If no file is given, or if it is −, the standard input is read.

USAGE

The m4 utility makes available the following built-in macros. These macros may be redefined, but once this is done the original meaning is lost. Their values are NULL unless otherwise stated.

changequote  Change quote symbols to the first and second arguments. The symbols may be up to five characters long. changequote without arguments restores the original values (that is, ‘ ’).

changecom    Change left and right comment markers from the default # and NEWLINE. With no arguments, the comment mechanism is effectively disabled. With one argument, the left marker becomes the argument and the right marker becomes NEWLINE. With two arguments, both markers are affected. Comment markers may be up to five characters long.

decr          Returns the value of its argument decremented by 1.

define        The second argument is installed as the value of the macro whose name is the first argument. Each occurrence of $n in the replacement text, where n is a digit, is replaced by the n-th argument. Argument 0 is the name of the macro; missing arguments are replaced by the null string; $# is replaced by the number of arguments; $* is replaced by a list of all the arguments separated by commas; $# is like $*, but each argument is quoted (with the current quotes).

defn          Returns the quoted definition of its argument(s). It is useful for renaming macros, especially built-ins.

divert        m4 maintains 10 output streams, numbered 0-9. The final output is the concatenation of the streams in numerical order; initially stream 0 is the current stream. The divert macro changes the current output stream to its (digit-string) argument. Output diverted to a stream other than 0 through 9 is discarded.

divnum        Returns the value of the current output stream.

dnl           Reads and discards characters up to and including the next NEWLINE.
dumpdef  Prints current names and definitions, for the named items, or for all if no arguments are given.

errprint  Prints its argument on the diagnostic output file.

/usr/ccs/bin/m4  eval  Evaluates its argument as an arithmetic expression, using 32-bit signed-integer arithmetic. The following operators are supported: parentheses, unary -, unary +, !, -, *, /, %, +, -, relational operators, bitwise & and ||. Octal and hex numbers may be specified as in C. The second argument specifies the radix for the result; the default is 10. The third argument may be used to specify the minimum number of digits in the result.

/usr/xpg4/bin/m4  eval  Evaluates its argument as an arithmetic expression, using 32-bit signed-integer arithmetic. The following operators are supported: parentheses, unary -, unary +, !, -, *, /, %, +, -, relational operators, bitwise & and ||. Precedence and associativity are as in C. Octal and hex numbers may also be specified as in C. The second argument specifies the radix for the result; the default is 10. The third argument may be used to specify the minimum number of digits in the result.

ifdef  If the first argument is defined, the value is the second argument, otherwise the third. If there is no third argument, the value is NULL. The word unix is predefined.

ifelse  This macro has three or more arguments. If the first argument is the same string as the second, then the value is the third argument. If not, and if there are more than four arguments, the process is repeated with arguments 4, 5, 6 and 7. Otherwise, the value is either the fourth string, or, if it is not present, NULL.

include  Returns the contents of the file named in the argument.

incr  Returns the value of its argument incremented by 1. The value of the argument is calculated by interpreting an initial digit-string as a decimal number.

index  Returns the position in its first argument where the second argument begins (zero origin), or -1 if the second argument does not occur.

len  Returns the number of characters in its argument.

m4exit  This macro causes immediate exit from m4. Argument 1, if given, is the exit code; the default is 0.

m4wrap  Argument 1 will be pushed back at final EOF; example: m4wrap('cleanup( )')

maketemp  Fills in a string of “X” characters in its argument with the current process ID.
popdef  Removes current definition of its argument(s), exposing the previous one, if any.
pushdef  Like define, but saves any previous definition.
shift   Returns all but its first argument. The other arguments are quoted and pushed back with commas in between. The quoting nullifies the effect of the extra scan that will subsequently be performed.
sinclude This macro is identical to include, except that it says nothing if the file is inaccessible.
substr  Returns a substring of its first argument. The second argument is a zero origin number selecting the first character; the third argument indicates the length of the substring. A missing third argument is taken to be large enough to extend to the end of the first string.
syscmd  This macro executes the command given in the first argument. No value is returned.
sysval  This macro is the return code from the last call to syscmd.
translit Transliterate the characters in its first argument from the set given by the second argument to the set given by the third. No abbreviations are permitted.
traceon This macro with no arguments, turns on tracing for all macros (including built-ins). Otherwise, turns on tracing for named macros.
traceoff Turns off trace globally and for any macros specified. Macros specifically traced by traceon can be untraced only by specific calls to traceoff.
undefined Removes the definition of the macro named in its argument.
undivert This macro causes immediate output of text from diversions named as arguments, or all diversions if no argument. Text may be undiverted into another diversion. Undiverting discards the diverted text.

EXAMPLES

EXAMPLE 1 Examples of m4 files.

An example of a single m4 input file capable of generating two output files follows. The file file1.m4 could contain lines such as:

if(VER, 1, do_something)
if(VER, 2, do_something)

The makefile for the program might include:

file1.1.c : file1.m4
    m4 -D VER=1 file1.m4 > file1.1.c
...
file1.2.c : file1.m4
EXAMPLE 1 Examples of m4 files.  (Continued)

    m4 -D VER=2 file1.m4 > file1.2.c
    ...

    m4 -D VER=1 file1.m4 > file1.1.c
    ...

    m4 -D VER=0 file1.m4 > file1.0.c
    ...

The -U option can be used to undefine VER. If file1.m4 contains:

    if(VER, 1, do_something)
    if(VER, 2, do_something)
    ifndef(VER, do_something)

then the makefile would contain:

    file1.0.c :  file1.m4
                m4 -U VER file1.m4 > file1.0.c
                ...

    file1.1.c :  file1.m4
                m4 -D VER=1 file1.m4 > file1.1.c
                ...

    file1.2.c :  file1.m4
                m4 -D VER=2 file1.m4 > file1.2.c
                ...

See environ(5) for descriptions of the following environment variables that affect the
execution of m4: LC_CTYPE, LC_MESSAGES, and NLSPATH.

The following exit values are returned:

    0  Successful completion.
    >0  An error occurred

If the m4exit macro is used, the exit value can be specified by the input file.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>/usr/ccs/bin/m4</th>
<th>attribute type</th>
<th>attribute value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>/usr/xpg4/bin/m4</th>
<th>attribute type</th>
<th>attribute value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWxcu4</td>
<td></td>
</tr>
</tbody>
</table>

SEE ALSO

as(1), attributes(5), environ(5), XPG4(5)
mach(1)

NAME mach – display the processor type of the current host

SYNOPSIS mach

DESCRIPTION The mach command displays the processor-type of the current host.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsus</td>
</tr>
</tbody>
</table>

SEE ALSO arch(1), uname(1), attributes(5)

NOTES mach and uname -p return equivalent values; therefore, Independent Software Vendors (ISV) and others who need to ascertain processor type are encouraged to use uname with the -p option instead of the mach command. The mach command is provided for compatibility with previous releases, but generally its use is discouraged.
NAME  | machid, sun, iAPX286, i286, i386, i486, i860, pdp11, sparc, u3b, u3b2, u3b5, u3b15, vax, u370 – get processor type truth value

SYNOPSIS  | sun  
           | iAPX286  
           | i386  
           | pdp11  
           | sparc  
           | u3b  
           | u3b2  
           | u3b5  
           | u3b15  
           | vax  
           | u370

DESCRIPTION | The following commands will return a true value (exit code of 0) if you are using an instruction set that the command name indicates.  
sun | True if you are on a Sun system.  
iAPX286 | True if you are on a computer using an iAPX286 processor.  
i386 | True if you are on a computer using an iAPX386 processor.  
pdp11 | True if you are on a PDP-11/45™ or PDP-11/70™.  
sparc | True if you are on a computer using a SPARC-family processor.  
u3b | True if you are on a 3B20 computer.  
u3b2 | True if you are on a 3B2 computer.  
u3b5 | True if you are on a 3B5 computer.  
u3b15 | True if you are on a 3B15 computer.  
vax | True if you are on a VAX-11/750™ or VAX-11/780™.  
u370 | True if you are on an IBM® System/370™ computer.

The commands that do not apply will return a false (non-zero) value. These commands are often used within makefiles (see make(1S)) and shell scripts (see sh(1)) to increase portability.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:
The machid family of commands is obsolete. Use `uname -p` and `uname -m` instead.

**SEE ALSO**
- `make(1S)`, `sh(1)`, `test(1)`, `true(1)`, `uname(1)`, `attributes(5)`

**NOTES**
The `machid` family of commands is obsolete. Use `uname -p` and `uname -m` instead.
mail, Mail – interactive message processing system

SYNOPSIS
/usr/ucb/mail ...
/usr/ucb/Mail ...

DESCRIPTION
/usr/ucb/mail and /usr/ucb/Mail are provided as links to /usr/bin/mailx. See mailx(1) for more information on the usage of these commands.

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

<table>
<thead>
<tr>
<th>PATH</th>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>/usr/ucb/mail</td>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
<tr>
<td>/usr/ucb/Mail</td>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

SEE ALSO
mailx(1), attributes(5)
mailcompat - provide SunOS compatibility for Solaris mailbox format

mailcompat is a program to provide SunOS 4.x compatibility for the Solaris mailbox format. You would typically run mailcompat to be able to read mail on a workstation running SunOS 4.x when your mail server is running Solaris.

Enabling mailcompat creates an entry in your .forward file, if it exists. If this file does not exist, mailcompat will create it. Disabling mailcompat will remove the entry from the .forward file, and if this was the only entry, will remove the entire file.

To execute mailcompat, log onto the Solaris mail server and enter mailcompat on the command line. Answer the queries provided by the program.

See largefile(5) for the description of the behavior of mailcompat when encountering files greater than or equal to 2 Gbyte (2^31 bytes).

EXAMPLE 1 Examples of the mailcompat feature.

The following example enables the mailcompat feature for the user "john".

eample% mailcompat
This program can be used to store your mail in a format that you can read with SunOS 4.X based mail readers
To enable the mailcompat feature a ".forward" file is created.
Would you like to enable the mailcompat feature? Y
Mailcompat feature ENABLED. Run mailcompat with no arguments to remove it example%

The following example disables the mailcompat feature for the user "john".

eample% mailcompat
This program can be used to store your mail in a format that you can read with SunOS 4.X based mail readers
You have a .forward file in your home directory containing:
"/usr/bin/mailcompat johns"
Would you like to remove it and disable the mailcompat feature? y
Back to normal reception of mail.
exaexample%

FILES ~/.forward list of recipients for forwarding messages

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO mailx(1), attributes(5), largefile(5)
mailp(1)

NAME
mailp, digestp, filep, newsp, filofaxp, franklinp, timemanp, timesysp – frontends to the
mp (Text to PDL (Printer Description Language) pretty print filter

SYNOPSIS
mailp [options] filename...
newsp [options] filename...
digestp [options] filename...
filep [options] filename...
filofaxp [options] filename...
franklinp [options] filename...
timemanp [options] filename...
timesysp [options] filename...

DESCRIPTION
The mailp utility is a frontend to the mp(1) program. It uses different names to
provide various mp options:

mailp                  Prints out mail messages.
newsp                  Prints out USENET news articles.
digestp                Prints out USENET digest files.
filep                  Prints out ordinary ASCII files.
filofaxp               Prints out in Filofax personal organiser format.
franklinp              Prints out in Franklin Planner personal organiser format.
timemanp               Prints out in Time Manager personal organiser format.
timesysp               Prints out in Time/System International personal organiser format.

mailp and the associated programs read each filename in sequence and generate a
prettified version of the contents. If no filename arguments are provided, mailp reads
the standard input.

mailp works in two ways. With the -D option, it will work as an X print server client
to produce the PDL of the target printer and spool it. With the -d or -P option, it will
generate and spool PostScript™ output.

OPTIONS
The following options are supported:

-\d printer            Sends output to the named printer. Otherwise, sends output to the
                        printer named in the PRINTER environment variable.
-\D
                        Generates the PDL for the target printer and spools it to the
                        printer.
-\F
                        Instead of printing who the mail article is for, the top header will
                        contain who the mail article is from. This is a useful option for
                        people with their own personal printer.
mailp(1)

- `h` Banner printing is disabled. Most of the information that typically appears on the banner sheet is output in the mp banners.
- `l` Formats output in landscape mode. Two pages of text will be printed per sheet of paper.
- `p printer` Same as `-d` option.
- `s subject` Uses `subject` as the new subject for the printout. If you are printing ordinary ASCII files which have been specified on the command line, the subject will default to the name of each of these files.

**OPERANDS**
The following operand is supported:

```plaintext
filename
```
The name of the file to be read.

**ENVIRONMENT VARIABLES**
If none of the `-d`, `-D`, or `-P` options is used, mailp uses the PRINTER environment variable to determine the printer to which the output from the mp(1) program is sent. If the PRINTER variable is not found, the default destination is the PostScript™ printer.

**EXIT STATUS**
The following exit values are returned:

<table>
<thead>
<tr>
<th>Exit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>1</td>
<td>An error occurred.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWmp</td>
</tr>
</tbody>
</table>

**SEE ALSO**
mp(1), attributes(5)

**NOTES**
The `-P` option, which spools the PDL directly to the target printer in mp(1), produces PostScript™ when used in mailp so as to be backward compatible.
NAME mailq – print the mail queue

SYNOPSIS /usr/bin/mailq [-q subarg] [-v]

DESCRIPTION The mailq utility displays a summary of the mail messages queued for future delivery.

The first line displayed for each mail message shows the internal identifier used on this host for the message, the size of the message in bytes, the date and time the message was accepted into the queue, and the envelope sender of the message. The second line of the display shows the error message that caused this message to be retained in the queue. This line will not be displayed if the message is being processed for the first time.

The mailq utility used to be identical to sendmail -bp. Now it checks for the authorization attribute, solaris.mail.mailq. If the check for the invoking user succeeds, sendmail -bp is executed with the remaining argument vector. Otherwise, an error message is printed. This authorization attribute is by default enabled for all users. It can be disabled by modifying the Basic Solaris User entry in prof_attr(4).

OPTIONS The following options are supported:

-qp[time] Similar to -q time, except that instead of periodically forking a child to process the queue, sendmail forks a single persistent child for each queue that alternates between processing the queue and sleeping. The sleep time is given as the argument. The sleep time default is 1 second. The process will always sleep at least 5 seconds if the queue was empty in the previous queue run.

-qf Processes saved messages in the queue once and does not fork(), but runs in the foreground.

-qG name Processes jobs in the queue group called name only.

-q[!]I substr Limits processed jobs to those containing substr as a substring of the queue id, or not when ! is specified.

-q[!]R substr Limits processed jobs to those containing substr as a substring of one of the recipients, or not when ! is specified.

-q[!]S substr Limits processed jobs to those containing substr as a substring of the sender, or not when ! is specified.

-v Prints verbose information. This adds the priority of the message and a single character indicator (+ or blank) indicating whether a warning message has been sent on the first line of the message. Additionally, extra lines may be intermixed with the recipients that indicate the "controlling user" information. This shows who will own any programs that are executed on behalf of this message and the name of the alias this command is expanded from, if any.

EXIT STATUS

| 0 | Successful completion. |
mailq(1)

>0 An error occurred.

FILES /etc/security/prof_attr

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsndmu</td>
</tr>
</tbody>
</table>

SEE ALSO sendmail(1M), prof_attr(4), attributes(5)
mailstats(1)

NAME
mailstats – print statistics collected by sendmail

SYNOPSIS

DESCRIPTION
The mailstats utility prints out the statistics collected by the sendmail(1M) program on mailer usage. These statistics are collected if the file indicated by the StatusFile configuration option of sendmail (defined in /etc/mail/sendmail.cf) exists. The default statistics file is /etc/mail/statistics.

To enable mailstats, you must, as root, touch /etc/mail/sendmail.st. See the StatusFile processing option in sendmail(1M).

mailstats first prints the time that the statistics file was created and the last time it was modified. Then, the statistics for each mailer are displayed on a single line, each with the following whitespace-separated fields:

- **M**  The mailer number.
- **msgsfr**  Number of messages from the mailer.
- **bytes_from**  Kbytes from the mailer.
- **msgsto**  Number of messages to the mailer.
- **bytes_to**  Kbytes to the mailer.
- **msgsrej**  Number of messages rejected by the mailer.
- **msgsdis**  Number of messages discarded by the mailer.
- **Mailer**  The name of the mailer.

After this display, a line totaling the values for all of the mailers is displayed, separated from the previous information by a line containing only equal sign (=) characters.

To reinitialize the statistics file once a night, add an entry to root’s crontab(1):

```
mailstats -p > /dev/null
```

OPTIONS
The following options are supported:

- **-c configfile**  Specifies a sendmail configuration file.
- **-C configfile**  Specifies a sendmail configuration file.
- **-f statisticsfile**  Specifies a sendmail statistics file.
- **-o**  Does not display the name of the mailer in the output.
- **-p**  Outputs information in program-readable mode and clear statistics.
- **-P**  Outputs information in program-readable mode without clearing statistics.

User Commands 795
mailstats(1)

**USAGE**
See largefile(5) for the description of the behavior of mailstats when encountering files greater than or equal to 2 Gbyte ($2^{31}$ bytes).

**FILES**
/dev/null
zero-lined file
/etc/mail/statistics
default sendmail statistics file
/etc/mail/sendmail.cf
default sendmail configuration file

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsndmu</td>
</tr>
</tbody>
</table>

**SEE ALSO**
crontab(1), cron(1M), sendmail(1M), attributes(5), largefile(5)
mailx(1)

NAME
mailx, mail – interactive message processing system

SYNOPSIS
mailx [-BdHiNURvV] [-f [file | +folder]] [-T file] [-u user]

mailx [-BdFintUvV] [-b bcc] [-c cc] [-h number] [-r address] [-s subject]

recipient...

/mail.../usr/ucb/mail...

/usr/ucb/Mail...

DESCRIPTION
The mail utilities listed above provide a comfortable, flexible environment for sending
and receiving mail messages electronically.

When reading mail, the mail utilities provide commands to facilitate saving, deleting,
and responding to messages. When sending mail, the mail utilities allow editing,
reviewing and other modification of the message as it is entered.

Incoming mail is stored in a standard file for each user, called the mailbox for that
user. When the mail utilities are called to read messages, the mailbox is the default
place to find them. As messages are read, they are marked to be moved to a secondary
file for storage, unless specific action is taken, so that the messages need not be seen
again. This secondary file is called the mbox and is normally located in the user’s HOME
directory (see MBOX in ENVIRONMENT VARIABLES for a description of this file).
Messages can be saved in other secondary files named by the user. Messages remain in
a secondary file until forcibly removed.

The user can access a secondary file by using the -f option. Messages in the
secondary file can then be read or otherwise processed using the same Commands as
in the primary mailbox. This gives rise within these pages to the notion of a current
mailbox.

OPTIONS
On the command line options start with a dash (-). Any other arguments are taken to
be destinations (recipients). If no recipients are specified, mailx attempts to read
messages from the mailbox.

-B
Do not buffer standard input or standard output.

-b bcc
Set the blind carbon copy list to bcc. bcc should be enclosed in
quotes if it contains more than one name.

-c cc
Set the carbon copy list to cc. cc should be enclosed in quotes if it
contains more than one name.

-d
Turn on debugging output. (Neither particularly interesting nor
recommended.)

-e
Test for the presence of mail. mailx prints nothing and exits with
a successful return code if there is mail to read.

-F
Record the message in a file named after the first recipient.
Overrides the record variable, if set (see Internal
Variables).
mailx(1)

-\ f \ [\ file] \ Read \ messages \ from \ file \ instead \ of \ mailbox. \ If \ no \ file \ is \ specified, \ the \ mailbox \ is \ used.
-\ f \ [\ \ -folder] \ Use \ the \ file \ folder \ in \ the \ folder \ directory \ (same \ as \ the \ folder \ command). \ The \ name \ of \ this \ directory \ is \ listed \ in \ the \ folder \ variable.
-\ H \ Print \ header \ summary \ only.
-\ h \ number \ The \ number \ of \ network \ “hops” \ made \ so \ far. \ This \ is \ provided \ for \ network \ software \ to \ avoid \ infinite \ delivery \ loops. \ This \ option \ and \ its \ argument \ are \ passed \ to \ the \ delivery \ program.
-\ I \ Include the newsgroup and article-id header lines when printing mail messages. This option requires the -f option to be specified.
-\ i \ Ignore interrupts. See also ignore in Internal Variables.
-\ N \ Do not print initial header summary.
-\ n \ Do not initialize from the system default mailx.rc or Mail.rc file. See USAGE.
-\ r \ address \ Use address as the return address when invoking the delivery program. All tilde commands are disabled. This option and its argument is passed to the delivery program.
-\ s \ subject \ Set the Subject header field to subject. subject should be enclosed in quotes if it contains embedded white space.
-\ T \ file \ Message-id and article-id header lines are recorded in file after the message is read. This option also sets the -I option.
-\ t \ Scan the input for To:, Cc:, and Bcc: fields. Any recipients on the command line will be ignored.
-\ U \ Convert UUCP-style addresses to internet standards. Overrides the conv environment variable.
-\ u \ user \ Read user's mailbox. This is only effective if user's mailbox is not read protected.
-\ V \ Print the mailx version number and exit.
-\ v \ Pass the -v flag to sendmail(1M).
-- \ Interpret tilde escapes in the input even if not reading from a tty.

OPERANDS
The following operands are supported:

recipient \ Addressee of message.

Starting Mail
At startup time, mailx executes the system startup file /etc/mail/mailx.rc. If invoked as mail or Mail, the system startup file /etc/mail/Mail.rc is used instead.

798 \ man pages section 1: User Commands • Last Revised 19 Sep 2001
The system startup file sets up initial display options and alias lists and assigns values to some internal variables. These variables are flags and valued parameters which are set and cleared using the `set` and `unset` commands. See `Internal Variables`.

With the following exceptions, regular commands are legal inside startup files: `!`, `Copy`, `edit`, `followup`, `Followup`, `hold`, `preserve`, `reply`, `Reply`, `shell`, and `visual`. An error in the startup file causes the remaining lines in the file to be ignored.

After executing the system startup file, the mail utilities execute the optional personal startup file `$HOME/.mailrc`, wherein the user can override the values of the internal variables as set by the system startup file.

If the `-n` option is specified, however, the mail utilities do not execute the system startup file.

Many system administrators include the commands

```
set appenddeadletter
unset replyall
unset pipeignore
```

in the system startup files (to be compatible with past Solaris behavior), but this does not meet standards requirements for `mailx`. To get standard behavior for `mailx`, users should use the `-n` option or include the following commands in a personal startup file:

```
unset appenddeadletter
set replyall
set pipeignore
```

When reading mail, the mail utilities are in `command mode`. A header summary of the first several messages is displayed, followed by a prompt indicating the mail utilities can accept regular commands (see `Commands` below). When sending mail, the mail utilities are in `input mode`. If no subject is specified on the command line, and the `asksub` variable is set, a prompt for the subject is printed.

As the message is typed, the mail utilities read the message and store it in a temporary file. Commands may be entered by beginning a line with the tilde (~) escape character followed by a single command letter and optional arguments. See `Tilde Escapes` for a summary of these commands.

### Reading Mail

Each message is assigned a sequential number, and there is at any time the notion of a current message, marked by a right angle bracket (>) in the header summary. Many commands take an optional list of messages (message-list) to operate on. In most cases, the current message is set to the highest-numbered message in the list after the command is finished executing.

The default for message-list is the current message. A message-list is a list of message identifiers separated by spaces, which may include:
mailx(1)

- Message number n.
- The current message.
^ The first undeleted message.
$ The last message.
* All messages.
+ The next undeleted message.
- The previous undeleted message.

n–m An inclusive range of message numbers.

user All messages from user.

/string All messages with string in the Subject line (case ignored).

:c All messages of type c, where c is one of:
  
  d  deleted messages
  n  new messages
  o  old messages
  r  read messages
  u  unread messages

Notice that the context of the command determines whether this type of message specification makes sense.

Other arguments are usually arbitrary strings whose usage depends on the command involved. Filenames, where expected, are expanded using the normal shell conventions (see sh(1)). Special characters are recognized by certain commands and are documented with the commands below.

Sending Mail

Recipients listed on the command line may be of three types: login names, shell commands, or alias groups. Login names may be any network address, including mixed network addressing. If mail is found to be undeliverable, an attempt is made to return it to the sender’s mailbox. If the recipient name begins with a pipe symbol ( | ), the rest of the name is taken to be a shell command to pipe the message through.

This provides an automatic interface with any program that reads the standard input, such as lp(1) for recording outgoing mail on paper. Alias groups are set by the alias command (see Commands below) or in a system startup file (for example, $HOME/.mailrc). Aliases are lists of recipients of any type.

Forwarding Mail

To forward a specific message, include it in a message to the desired recipients with the -f or -m tilde escapes. See Tilde Escapes below. To forward mail automatically, add a comma-separated list of addresses for additional recipients to the .forward file in your home directory. This is different from the format of the alias command,
which takes a space-separated list instead. Note: Forwarding addresses must be valid, or the messages will “bounce.” You cannot, for instance, reroute your mail to a new host by forwarding it to your new address if it is not yet listed in the NIS aliases domain.

**Commands**

Regular commands are of the form

```
[ command ] [ message-list ] [ arguments ]
```

In *input mode*, commands are recognized by the escape character, `~`, and lines not treated as commands are taken as input for the message. If no command is specified in *command mode*, next is assumed. The following is a complete list of *mailx* commands:

1. **shell-command**
   - Escape to the shell. See `SHELL` in Environment Variables.

2. **# comment**
   - NULL command (comment). Useful in *mailrc* files.

3. **=`
   - Print the current message number.

4. **?**
   - Prints a summary of commands.

5. **alias alias name . . .**

6. **group alias name . . .**
   - Declare an alias for the given names. The names are substituted when `alias` is used as a recipient. Useful in the *mailrc* file. With no arguments, the command displays the list of defined aliases.

7. **alternates name . . .**
   - Declare a list of alternate names for your login. When responding to a message, these names are removed from the list of recipients for the response. With no arguments, print the current list of alternate names. See also `allnet` in Internal Variables.

8. **cd [directory]**

9. **chdir [directory]**
   - Change directory. If `directory` is not specified, `$HOME` is used.

10. **copy [file]**

11. **copy [message-list] file**
    - Copy messages to the file without marking the messages as saved. Otherwise equivalent to the `save` command.

12. **Copy [message-list]**
    - Save the specified messages in a file whose name is derived from the author of the message to be saved, without marking the messages as saved. Otherwise equivalent to the `save` command.
delete [message-list]
   Delete messages from the mailbox. If autoprint is set, the next message after the last one deleted is printed (see Internal Variables).

discard [header-field...]
ignore [header-field...]
   Suppress printing of the specified header fields when displaying messages on the screen. Examples of header fields to ignore are Status and Received. The fields are included when the message is saved, unless the alwaysignore variable is set. The More, Page, Print, and Type commands override this command. If no header is specified, the current list of header fields being ignored is printed. See also the undiscard and unignore commands.

dp [message-list]
dt [message-list]
   Delete the specified messages from the mailbox and print the next message after the last one deleted. Roughly equivalent to a delete command followed by a print command.

echo string ...
   Echo the given strings (like echo(1)).

edit [message-list]
   Edit the given messages. Each message is placed in a temporary file and the program named by the EDITOR variable is invoked to edit it (see ENVIRONMENT VARIABLES). Default editor is ed(1).

exit
exit
   Exit from mailx, without changing the mailbox. No messages are saved in the mbox (see also quit).

field [message-list] header-file
   Display the value of the header field in the specified message.

file [file]
folder [file]
   Quit from the current file of messages and read in the specified file. Several special characters are recognized when used as file names:

   % the current mailbox.
   %user the mailbox for user.
   # the previous mail file.
   & the current mbox.
   +file The named file in the folder directory (listed in the folder variable).

With no arguments, print the name of the current mail file, and the number of messages and characters it contains.
folders
Print the names of the files in the directory set by the folder variable (see Internal Variables).

Followup [message]
Respond to a message, recording the response in a file whose name is derived from the author of the message. Overrides the record variable, if set. If the replyall variable is set, the actions of Followup and followup are reversed. See also the followup, Save, and Copy commands and outfolder in Internal Variables, and the Starting Mail section in USAGE above.

followup [message-list]
Respond to the first message in the message-list, sending the message to the author of each message in the message-list. The subject line is taken from the first message and the response is recorded in a file whose name is derived from the author of the first message. If the replyall variable is set, the actions of followup and Followup are reversed. See also the Followup, Save, and Copy commands and outfolder in Internal Variables, and the Starting Mail section in USAGE above.

from [message-list]
Print the header summary for the specified messages. If no messages are specified, print the header summary for the current message.

group alias name . . .
alias alias name . . .
Declare an alias for the given names. The names are substituted when alias is used as a recipient. Useful in the mailrc file.

headers [message]
Print the page of headers which includes the message specified. The screen variable sets the number of headers per page (see Internal Variables). See also the z command.

help
Print a summary of commands.

hold [message-list]
preserve [message-list]
Hold the specified messages in the mailbox.

if s | r | t
mail-commands
else
mail-commands
endif
Conditional execution, where s executes following mail-commands, up to an else or endif, if the program is in send mode, r causes the mail-commands to be executed only in receive mode, and t causes the mail-commands to be executed only if mailx is being run from a terminal. Useful in the mailrc file.
inc
Incorporate messages that arrive while you are reading the system mailbox. The 
new messages are added to the message list in the current mail session. This 
command does not commit changes made during the session, and prior messages 
are not renumbered.

ignore [header-field ...]
discard [header-field ...]
Suppress printing of the specified header fields when displaying messages on the 
screen. Examples of header fields to ignore are Status and Cc. All fields are 
included when the message is saved. The More, Page, Print and Type commands 
override this command. If no header is specified, the current list of header fields 
being ignored is printed. See also the undiscard and unignore commands.

list
Print all commands available. No explanation is given.

load
[message] file The specified message is replaced by the message in the named file. 
file should contain a single mail message including mail headers (as saved by the 
save command).

mail recipient ...
Mail a message to the specified recipients.

Mail recipient
Mail a message to the specified recipients, and record it in a file whose name is 
derived from the author of the message. Overrides the record variable, if set. See 
also the Save and Copy commands and outfolder in Internal Variables.

mbox [message-list]
Arrange for the given messages to end up in the standard mbox save file when 
mailx terminates normally. See MBOX in ENVIRONMENT VARIABLES for a 
description of this file. See also the exit and quit commands.

more [message-list]
page [message-list]
Print the specified messages. If crt is set, the messages longer than the number of 
lines specified by the crt variable are paged through the command specified by the 
PAGER variable. The default command is pg(1) or if the bsdcompat variable is set, 
the default is more(1). See ENVIRONMENT VARIABLES. Same as the print and 
type commands.

More [message-list]
Page [message-list]
Print the specified messages on the screen, including all header fields. Overrides 
suppression of fields by the ignore command. Same as the Print and Type 
commands.

new [message-list]
New [message-list]
unread [message-list]
Unread

[message-list] Take a message list and mark each message as not having been read.

next [message]

Go to the next message matching message. If message is not supplied, this command finds the next message that was not deleted or saved. A message-list may be specified, but in this case the first valid message in the list is the only one used. This is useful for jumping to the next message from a specific user, since the name would be taken as a command in the absence of a real command. See the discussion of message-list above for a description of possible message specifications.

pipe [message-list] [shell-command]

Pipe the message through the given shell-command. The message is treated as if it were read. If no arguments are given, the current message is piped through the command specified by the value of the cmd variable. If the page variable is set, a form feed character is inserted after each message (see Internal Variables).

preserve [message-list]

hold [message-list]

Preserve the specified messages in the mailbox.

print [message-list]

type [message-list]

Print the specified messages. If crl is set, the messages longer than the number of lines specified by the crl variable are paged through the command specified by the PAGER variable. The default command is pg(1) or if the bsdcompat variable is set, the default is more(1). See ENVIRONMENT VARIABLES. Same as the more and page commands.

Print [message-list]

Type [message-list]

Print the specified messages on the screen, including all header fields. Overrides suppression of fields by the ignore command. Same as the More and Page commands.

put [file]

put [message-list] file

Save the specified message in the given file. Use the same conventions as the print command for which header fields are ignored.

Put [file]

Put [message-list] file

Save the specified message in the given file. Overrides suppression of fields by the ignore command.

quit

Exit from mailx, storing messages that were read in mbox and unread messages in the mailbox. Messages that have been explicitly saved in a file are deleted unless the keepsave variable is set.
Send a response to the author of each message in the message-list. The subject line is taken from the first message. If record is set to a file, a copy of the reply is added to that file. If the replyall variable is set, the actions of Reply/Respond and reply/respond are reversed. The replysender command is not affected by the replyall variable, but sends each reply only to the sender of each message. See the Starting Mail section in USAGE above.

Reply [message]
Respond [message]
replyall [message]

Reply to the specified message, including all other recipients of that message. If the variable record is set to a file, a copy of the reply added to that file. If the replyall variable is set, the actions of Reply/Respond and reply/respond are reversed. The replyall command is not affected by the replyall variable, but always sends the reply to all recipients of the message. See the Starting Mail section in USAGE above.

retain

Add the list of header fields named to the retained list. Only the header fields in the retain list are shown on your terminal when you print a message. All other header fields are suppressed. The set of retained fields specified by the retain command overrides any list of ignored fields specified by the ignore command. The Type and Print commands can be used to print a message in its entirety. If retain is executed with no arguments, it lists the current set of retained fields.

Save [message-list]

Save the specified messages in a file whose name is derived from the author of the first message. The name of the file is taken to be the author’s name with all network addressing stripped off. See also the Copy, followup, and Followup commands and outfolder in Internal Variables.

save [file]

Save the specified messages in the given file. The file is created if it does not exist. The file defaults to mbox. The message is deleted from the mailbox when mailx terminates unless keepsave is set (see also Internal Variables and the exit and quit commands).

set

set variable
set variable=string
set variable=number

Define a variable. To assign a value to variable, separate the variable name from the value by an ‘=’ (there must be no space before or after the ‘=’). A variable may be given a null, string, or numeric value. To embed SPACE characters within a value, enclose it in quotes.
With no arguments, `set` displays all defined variables and any values they might have. See `Internal Variables` for a description of all predefined mail variables.

`shell`
- Invoke an interactive shell. See also `SHELL` in `ENVIRONMENT VARIABLES`.

`size [message-list]`
- Print the size in characters of the specified messages.

`sourcelfile`
- Read commands from the given file and return to command mode.

`top [message-list]`
- Print the top few lines of the specified messages. If the `toplines` variable is set, it is taken as the number of lines to print (see `Internal Variables`). The default is 5.

`touched [message-list]`
- Touch the specified messages. If any message in `message-list` is not specifically saved in a file, it is placed in the `mbox`, or the file specified in the `MBOX` environment variable, upon normal termination. See `exit` and `quit`.

`type [message-list]`
- Print `message-list` on the screen, including all header fields. Overrides suppression of fields by the `ignore` command.

`print [message-list]`
- Print the specified messages. If `crt` is set, the messages longer than the number of lines specified by the `crt` variable are paged through the command specified by the `PAGER` variable. The default command is `pg(1)`. See `ENVIRONMENT VARIABLES`.

`unalias [alias]...`
`ungroup [alias]...`
- Remove the definitions of the specified aliases.

`undeleter [message-list]`
- Restore the specified deleted messages. Will only restore messages deleted in the current mail session. If `autorestore` is set, the last message of those restored is printed (see `Internal Variables`).

`undiscard [header-field...]
unignore [header-field...]
unretaine [header-field...]
unread [message-list]`
- Remove the specified header fields from the list being ignored. If no header fields are specified, all header fields are removed from the list being ignored.

- Remove the specified header fields from the list being retained. If no header fields are specified, all header fields are removed from the list being retained.

`unread [message-list]`
`Unread [message-list]` Same as the new command.
 unset variable ...
   Erase the specified variables. If the variable was imported from the environment
   (that is, an environment variable or exported shell variable), it cannot be unset from
   within mailx.

 version
   Print the current version and release date of the mailx utility.

 visual [message-list]
   Edit the given messages with a screen editor. Each messages is placed in a
   temporary file and the program named by the VISUAL variable is invoked to edit it
   (see ENVIRONMENT VARIABLES). Notice that the default visual editor is vi.

 write [message-list] file
   Write the given messages on the specified file, minus the header and trailing blank
   line. Otherwise equivalent to the save command.

 xit
 exit
   Exit from mailx, without changing the mailbox. No messages are saved in the
   mbox (see also quit).

 z [+ | -]
   Scroll the header display forward or backward one screen–full. The number of
   headers displayed is set by the screen variable (see Internal Variables).

**Tilde Escapes**

The following tilde escape commands can be used when composing mail to send.
These may be entered only from input mode, by beginning a line with the tilde escape
character (~). See escape in Internal Variables for changing this special
character. The escape character can be entered as text by typing it twice.

- ! shell-command
  Escape to the shell. If present, run shell-command.

- .
  Simulate end of file (terminate message input).

- : mail-command
- _ mail-command
  Perform the command-level request. Valid only when
  sending a message while reading mail.

- ?
  Print a summary of tilde escapes.

- A
  Insert the autograph string Sign into the message (see
  Internal Variables).

- a
  Insert the autograph string sign into the message (see
  Internal Variables).

- b name ...
  Add the names to the blind carbon copy (Bcc) list. This
  is like the carbon copy (Cc) list, except that the names
  in the Bcc list are not shown in the header of the mail
  message.

- c name ...
  Add the names to the carbon copy (Cc) list.
-d Read in the dead-letter file. See DEAD in ENVIRONMENT VARIABLES for a description of this file.

-e Invoke the editor on the partial message. See also EDITOR in ENVIRONMENT VARIABLES.

-\f [message-list] Forward the specified message, or the current message being read. Valid only when sending a message while reading mail. The messages are inserted into the message without alteration (as opposed to the -m escape).

-\f [message-list] Forward the specified message, or the current message being read, including all header fields. Overrides the suppression of fields by the ignore command.

-h Prompt for Subject line and To, Cc, and Bcc lists. If the field is displayed with an initial value, it may be edited as if you had just typed it.

-\i variable Insert the value of the named variable into the text of the message. For example, -\A is equivalent to `'-i Sign.' Environment variables set and exported in the shell are also accessible by -\i.

-\m [message-list] Insert the listed messages, or the current message being read into the letter. Valid only when sending a message while reading mail. The text of the message is shifted to the right, and the string contained in the indentprefix variable is inserted as the leftmost characters of each line. If indentprefix is not set, a TAB character is inserted into each line.

-\M [message-list] Insert the listed messages, or the current message being read, including the header fields, into the letter. Valid only when sending a message while reading mail. The text of the message is shifted to the right, and the string contained in the indentprefix variable is inserted as the leftmost characters of each line. If indentprefix is not set, a TAB character is inserted into each line. Overrides the suppression of fields by the ignore command.

-p Print the message being entered.

-q Quit from input mode by simulating an interrupt. If the body of the message is not null, the partial message is saved in dead-letter. See DEAD in ENVIRONMENT VARIABLES for a description of this file.

-R Mark message for return receipt.
-r file
-
-ss file
-
-s shell-command

Read in the specified file. If the argument begins with an exclamation point (!), the rest of the string is taken as an arbitrary shell command and is executed, with the standard output inserted into the message.

-a string ...

Set the subject line to string.

-t name ...

Add the given names to the To list.

-v

Invoke a preferred screen editor on the partial message. The default visual editor is vi(1). See also VISUAL in ENVIRONMENT VARIABLES.

-w file

Write the message into the given file, without the header.

-x

Exit as with -q except the message is not saved in dead-letter.

-| shell-command

Pipe the body of the message through the given shell-command. If the shell-command returns a successful exit status, the output of the command replaces the message.

Internal Variables

The following variables are internal variables. They may be imported from the execution environment or set using the set command at any time. The unset command may be used to erase variables.

allnet

All network names whose last component (login name) match are treated as identical. This causes the message-list message specifications to behave similarly. Disabled by default. See also the alternates command and the metoo and fuzzymatch variables.

alwaysignore

Ignore header fields with ignore everywhere, not just during print or type. Affects the save, Save, copy, Copy, top, pipe, and write commands, and the -m and -f tilde escapes. Enabled by default.

append

Upon termination, append messages to the end of the mbox file instead of prepending them. Although disabled by default, append is set in the system startup file (which can be suppressed with the -n command line option).

appenddeadletter

Append to the deadletter file rather than overwrite it. Although disabled by default, appenddeadletter is frequently set in the system startup file. See Starting Mail in USAGE above.
askbcc
Prompt for the Bcc list after the Subject is entered if it is not specified on the command line with the -b option. Disabled by default.

askcc
Prompt for the Cc list after the Subject is entered if it is not specified on the command line with the -c option. Disabled by default.

asksub
Prompt for subject if it is not specified on the command line with the -s option. Enabled by default.

autoinc
Automatically incorporate new messages into the current session as they arrive. This has an affect similar to issuing the inc command every time the command prompt is displayed. Disabled by default, but autoinc is set in the default system startup file for mailx; it is not set for /usr/ucb/mail or /usr/ucb/Mail.

autoprint
Enable automatic printing of messages after delete and undelete commands. Disabled by default.

bang
Enable the special-casing of exclamation points (!) in shell escape command lines as in vi(1). Disabled by default.

bsdcompat
Set automatically if mailx is invoked as mail or Mail. Causes mailx to use /etc/mail/Mail.rc as the system startup file. Changes the default pager to more(1).

cmd=shell-command
Set the default command for the pipe command. No default value.

conv=conversion
Convert uucp addresses to the specified address style, which can be either:

internet
This requires a mail delivery program conforming to the RFC822 standard for electronic mail addressing.

optimize
Remove loops in uucp(1C) address paths (typically generated by the reply command). No rerouting is performed; mail has no knowledge of UUCP routes or connections.

Conversion is disabled by default. See also sendmail(1M) and the -U command-line option.

crt[number]
Pipe messages having more than number lines through the command specified by the value of the PAGER
variable (pg(1) or more(1) by default). If number is not specified, the current window size is used. Disabled by default.

debug
Enable verbose diagnostics for debugging. Messages are not delivered. Disabled by default.

dot
Take a period on a line by itself, or EOF during input from a terminal as end-of-file. Disabled by default, but dot is set in the system startup file (which can be suppressed with the -n command line option).

fcc
By default, mailx will treat any address containing a slash ("/") character as a local "send to file" address. By unsetting this option, this behavior is disabled. Enabled by default.

flipr
Reverse the effect of the followup/Followup and reply/Reply command pairs. If both flipr and replyall are set, the effect is as if neither was set.

from
Extract the author listed in the header summary from the From: header instead of the UNIX From line. Enabled by default.

fuzzymatch
The from command searches for messages from the indicated sender. By default, the full sender address must be specified. By setting this option, only a sub-string of the sender address need be specified. Disabled by default.

escape=c
Substitute c for the ~ escape character. Takes effect with next message sent.

folder=directory
The directory for saving standard mail files. User-specified file names beginning with a plus (+) are expanded by preceding the file name with this directory name to obtain the real file name. If directory does not start with a slash (/), $HOME is prepended to it. There is no default for the folder variable. See also outfolder below.

header
Enable printing of the header summary when entering mailx. Enabled by default.

hold
Preserve all messages that are read in the mailbox instead of putting them in the standard mbox save file. Disabled by default.

ignore
Ignore interrupts while entering messages. Handy for noisy dial-up lines. Disabled by default.
ignoreeof

Ignore end-of-file during message input. Input must be terminated by a period (.) on a line by itself or by the ~. command. See also dot above. Disabled by default.

indentprefix=string

When indentprefix is set, string is used to mark indented lines from messages included with -m. The default is a TAB character.

keep

When the mailbox is empty, truncate it to zero length instead of removing it. Disabled by default.

iprompt=string

The specified prompt string is displayed before each line on input is requested when sending a message.

keepsave

Keep messages that have been saved in other files in the mailbox instead of deleting them. Disabled by default.

makeremote

When replying to all recipients of a message, if an address does not include a machine name, it is assumed to be relative to the sender of the message. Normally not needed when dealing with hosts that support RFC822.

metoo

If your login appears as a recipient, do not delete it from the list. Disabled by default.

mustbang

Force all mail addresses to be in bang format.

onehop

When responding to a message that was originally sent to several recipients, the other recipient addresses are normally forced to be relative to the originating author’s machine for the response. This flag disables alteration of the recipients’ addresses, improving efficiency in a network where all machines can send directly to all other machines (that is, one hop away). Disabled by default.

outfolder

Locate the files used to record outgoing messages in the directory specified by the folder variable unless the path name is absolute. Disabled by default. See folder above and the Save, Copy, followup, and Followup commands.

page

Used with the pipe command to insert a form feed after each message sent through the pipe. Disabled by default.

pipeignore

Omit ignored header when outputting to the pipe command. Although disabled by default, pipeignore is frequently set in the system startup file. See Starting Mail in USAGE above.
postmark

Your "real name" to be included in the From line of messages you send. By default this is derived from the comment field in your passwd(4) file entry.

prompt=string

Set the command mode prompt to string. Default is "? ", unless the bsdcompat variable is set, then the default is "&".

quiet

Refrain from printing the opening message and version when entering mailx. Disabled by default.

record=file

Record all outgoing mail in file. Disabled by default. See also outfolder above.

replyall

Reverse the effect of the reply and Reply and Followup and Followup commands. Although set by default, replayall is frequently unset in the system startup file. See flipr and Starting Mail in USAGE above.

returnaddr=string

The default sender address is that of the current user. This variable can be used to set the sender address to any arbitrary value. Set with caution.

save

Enable saving of messages in dead-letter on interrupt or delivery error. See DEAD for a description of this file. Enabled by default.

screen=number

Sets the number of lines in a screen-full of headers for the headers command. number must be a positive number.

The default is set according to baud rate or window size. With a baud rate less than 1200, number defaults to 5, if baud rate is exactly 1200, it defaults to 10. If you are in a window, number defaults to the default window size minus 4. Otherwise, the default is 20.

sendmail=shell-command

Alternate command for delivering messages. Note: In addition to the expected list of recipients, mail also passes the -i and -m, flags to the command. Since these flags are not appropriate to other commands, you may have to use a shell script that strips them from the arguments list before invoking the desired command. Default is /usr/bin/rmail.

sendwait

Wait for background mailer to finish before returning. Disabled by default.

showname

Causes the message header display to show the sender's real name (if known) rather than their mail
address. Disabled by default, but showname is set in the /etc/mail/mailx.rc system startup file for mailx.

showto
When displaying the header summary and the message is from you, print the recipient’s name instead of the author’s name.

sign=string
The variable inserted into the text of a message when the -a (autograph) command is given. No default (see also -i in Tilde Escapes).

Sign=string
The variable inserted into the text of a message when the -A command is given. No default (see also -i in Tilde Escapes).

toplines=number
The number of lines of header to print with the top command. Default is 5.

verbose
Invoke sendmail(1M) with the -v flag.

translate
The name of a program to translate mail addresses. The program receives mail addresses as arguments. The program produces, on the standard output, lines containing the following data, in this order:

- the postmark for the sender (see the postmark variable)
- translated mail addresses, one per line, corresponding to the program’s arguments. Each translated address will replace the corresponding address in the mail message being sent.
- a line containing only "y" or "n". if the line contains "y" the user will be asked to confirm that the message should be sent.

The translate program will be invoked for each mail message to be sent. If the program exits with a non-zero exit status, or fails to produce enough output, the message is not sent.

See largefile(5) for the description of the behavior of mailx when encountering files greater than or equal to 2 Gbyte ($2^{31}$ bytes).

See environ(5) for descriptions of the following environment variables that affect the execution of mailx: HOME, LANG, LC_CTYPE, LC_TIME, LC_MESSAGES, NLSPATH, and TERM.

DEAD
The name of the file in which to save partial letters in case of untimely interrupt. Default is $HOME/dead.letter.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDITOR</td>
<td>The command to run when the <code>edit</code> or <code>-e</code> command is used. Default is <code>ed(1)</code>.</td>
</tr>
<tr>
<td>LISTER</td>
<td>The command (and options) to use when listing the contents of the folder directory. The default is <code>ls(1)</code>.</td>
</tr>
<tr>
<td>MAIL</td>
<td>The name of the initial mailbox file to read (in lieu of the standard system mailbox). The default is <code>/var/mail/username</code>.</td>
</tr>
<tr>
<td>MAILRC</td>
<td>The name of the startup file. Default is <code>$HOME/.mailrc</code>.</td>
</tr>
<tr>
<td>MAILX_HEAD</td>
<td>The specified string is included at the beginning of the body of each message that is sent.</td>
</tr>
<tr>
<td>MAILX_TAIL</td>
<td>The specified string is included at the end of the body of each message that is sent.</td>
</tr>
<tr>
<td>MBOX</td>
<td>The name of the file to save messages which have been read. The <code>exit</code> command overrides this function, as does saving the message explicitly in another file. Default is <code>$HOME/mbox</code>.</td>
</tr>
<tr>
<td>PAGER</td>
<td>The command to use as a filter for paginating output. This can also be used to specify the options to be used. Default is <code>pg(1)</code>, or if the <code>bsdcompat</code> variable is set, the default is <code>more(1)</code>. See <code>Internal Variables</code>.</td>
</tr>
<tr>
<td>SHELL</td>
<td>The name of a preferred command interpreter. Default is <code>sh(1)</code>.</td>
</tr>
<tr>
<td>VISUAL</td>
<td>The name of a preferred screen editor. Default is <code>vi(1)</code>.</td>
</tr>
</tbody>
</table>

**EXIT STATUS**

When the `-e` option is specified, the following exit values are returned:

- 0: Mail was found.
- >0: Mail was not found or an error occurred.

Otherwise, the following exit values are returned:

- 0: Successful completion. Notice that this status implies that all messages were sent, but it gives no assurances that any of them were actually delivered.
- >0: An error occurred

**FILES**

- `$HOME/.mailrc`
  - personal startup file
- `$HOME/mbox`
  - secondary storage file
- `$HOME/.Maillock`
  - lock file to prevent multiple writers of system mailbox
- `/etc/mail/mailx.rc`
  - optional system startup file for `mailx` only
/etc/mail/Mail.rc
  BSD compatibility system-wide startup file for /usr/ucb/mail and
  /usr/ucb/Mail
/tmp/R[emqsx]*
  temporary files
/usr/share/lib/mailx/mailx.help*
  help message files
/var/mail/*
  post office directory

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO
biff(1B), echo(1), ed(1), ex(1), fmt(1), lp(1), ls(1), mail(1), mail(1B),
mailcompat(1), more(1), pg(1), sh(1), uucp(1C), vacation(1), vi(1),
newaliases(1M), sendmail(1M), aliases(4), passwd(4), attributes(5),
environ(5), largefile(5), standards(5)

NOTES
Where shell-command is shown as valid, arguments are not always allowed.
Experimentation is recommended.

Internal variables imported from the execution environment cannot be unset.

The full internet addressing is not fully supported by mailx. The new standards need
some time to settle down.

Replies do not always generate correct return addresses. Try resending the errant reply
with onehop set.

mailx does not lock your record file. So, if you use a record file and send two or more
messages simultaneously, lines from the messages may be interleaved in the record
file.

The format for the alias command is a space-separated list of recipients, while the
format for an alias in either the .forward or /etc/aliases is a comma-separated
list.

To read mail on a workstation running Solaris 1.X when your mail server is running
Solaris 2.X, first execute the mailcompat(1) program.
make(1S)

NAME  make – maintain, update, and regenerate related programs and files

             [target...] [macro = value...]

             = value...]

DESCRIPTION  The make utility executes a list of shell commands associated with each target,
typically to create or update a file of the same name. makefile contains entries that
describe how to bring a target up to date with respect to those on which it depends,
which are called dependencies. Since each dependency is a target, it may have
dependencies of its own. Targets, dependencies, and sub-dependencies comprise a tree
structure that make traces when deciding whether or not to rebuild a target.

The make utility recursively checks each target against its dependencies, beginning
with the first target entry in makefile if no target argument is supplied on the command
line. If, after processing all of its dependencies, a target file is found either to be
missing, or to be older than any of its dependencies, make rebuilds it. Optionally with
this version of make, a target can be treated as out-of-date when the commands used
to generate it have changed since the last time the target was built.

To build a given target, make executes the list of commands, called a rule. This rule
may be listed explicitly in the target’s makefile entry, or it may be supplied implicitly
by make.

If no target is specified on the command line, make uses the first target defined in
makefile.

If a target has no makefile entry, or if its entry has no rule, make attempts to derive a
rule by each of the following methods, in turn, until a suitable rule is found. Each
method is described under USAGE below.

- Pattern matching rules.
- Implicit rules, read in from a user-supplied makefile.
- Standard implicit rules (also known as suffix rules), typically read in from the file
  /usr/share/lib/make/make.rules.
- SCCS retrieval. make retrieves the most recent version from the SCCS history file
  (if any). See the description of the .SCCS_GET: special-function target for details.
- The rule from the .DEFAULT: target entry, if there is such an entry in the makefile.

If there is no makefile entry for a target, if no rule can be derived for building it, and if
no file by that name is present, make issues an error message and halts.

OPTIONS  The following options are supported:
Displays the reasons why `make` chooses to rebuild a target; `make` displays any and all dependencies that are newer. In addition, `make` displays options read in from the `MAKEFLAGS` environment variable.

Displays the dependency check and processing in vast detail.

Displays the text of the `make` files read in.

Displays the text of the `make` files, `make.rules` file, the `state` file, and all hidden-dependency reports.

Environment variables override assignments within `make` files.

Uses the description file `makefile`. A `-'` as the `makefile` argument denotes the standard input. The contents of `makefile`, when present, override the standard set of implicit rules and predefined macros. When more than one `-f makefile` argument pair appears, `make` uses the concatenation of those files, in order of appearance.

When no `makefile` is specified, `/usr/ccs/bin/make` tries the following in sequence, except when in POSIX mode (see the .POSIX Special-Function Targets in the USAGE section below):

- If there is a file named `makefile` in the working directory, `make` uses that file. If, however, there is an SCCS history file (`SCCS/s.makefile`) which is newer, `make` attempts to retrieve and use the most recent version.
- In the absence of the above file(s), if a file named `Makefile` is present in the working directory, `make` attempts to use it. If there is an SCCS history file (`SCCS/s.Makefile`) that is newer, `make` attempts to retrieve and use the most recent version.

When no `makefile` is specified, `/usr/ccs/bin/make` in POSIX mode and `/usr/xpg4/bin/make` try the following files in sequence:

- `./makefile`, `./Makefile`
- `s.makefile`, `SCCS/s.makefile`
- `s.Makefile`, `SCCS/s.Makefile`

Ignores error codes returned by commands. Equivalent to the special-function target `.'IGNORE:'`.

When a nonzero error status is returned by a rule, or when `make` cannot find a rule, abandons work on the current target, but continues with other dependency branches that do not depend on it.

Uses the state file `statefile`. A `-'` as the `statefile` argument denotes the standard input. The contents of `statefile`, when present, override the
standard set of implicit rules and predefined macros. When more
than one `-K statefile' argument pair appears, make uses the
concatenation of those files, in order of appearance. (See also
`.KEEP_STATE and .KEEP_STATE_FILE in the Special-Function
Targets section).

-n  No execution mode. Prints commands, but does not execute them.
Even lines beginning with an @ are printed. However, if a
command line contains a reference to the $(MAKE) macro, that
line is always executed (see the discussion of MAKEFLAGS in
Reading Makefiles and the Environment). When in POSIX mode,
lines beginning with a “+” are executed.

-p  Prints out the complete set of macro definitions and target
descriptions.

-P  Merely reports dependencies, rather than building them.

-q  Question mode. make returns a zero or nonzero status code
depending on whether or not the target file is up to date. When in
POSIX mode, lines beginning with a “+” are executed.

-r  Does not read in the default makefile
/usr/share/lib/make/make.rules.

-s  Silent mode. Does not print command lines before executing them.
Equivalent to the special-function target .SILENT:

-S  Undoes the effect of the -k option. Stops processing when a
non-zero exit status is returned by a command.

-t  Touches the target files (bringing them up to date) rather than
performing their rules. This can be dangerous when files are
maintained by more than one person. When the .KEEP_STATE:
target appears in the makefile, this option updates the state file just
as if the rules had been performed. When in POSIX mode, lines
beginning with a “+” are executed.

-V  Puts make into SysV mode. Refer to sysV-make(1) for respective
details.

OPERANDS
The following operands are supported:

target  Target names, as defined in USAGE.

macro=value  Macro definition. This definition overrides any regular definition
for the specified macro within the makefile itself, or in the
environment. However, this definition can still be overridden by
conditional macro assignments.
When make first starts, it reads the MAKEFLAGS environment variable to obtain any of the following options specified present in its value: -d, -D, -e, -i, -k, -n, -p, -q, -r, -s, -S, or -t. Due to the implementation of POSIX.2 (see POSIX.2(5), the MAKEFLAGS values will contain a leading ‘−’ character. The make utility then reads the command line for additional options, which also take effect.

Next, make reads in a default makefile that typically contains predefined macro definitions, target entries for implicit rules, and additional rules, such as the rule for retrieving SCCS files. If present, make uses the file make.rules in the current directory; otherwise it reads the file /usr/share/lib/make/make.rules, which contains the standard definitions and rules. Use the directive:

```
include /usr/share/lib/make/make.rules
```
in your local make.rules file to include them.

Next, make imports variables from the environment (unless the -e option is in effect), and treats them as defined macros. Because make uses the most recent definition it encounters, a macro definition in the makefile normally overrides an environment variable of the same name. When −e is in effect, however, environment variables are read in after all makefiles have been read. In that case, the environment variables take precedence over definitions in the makefile.

Next, make reads any makefiles you specify with -f, or one of makefile or Makefile as described above and then the state file, in the local directory if it exists. If the makefile contains a .KEEP_STATE_FILE target, then it reads the state file that follows the target. Refer to special target .KEEP_STATE_FILE for details.

Next (after reading the environment if -e is in effect), make reads in any macro definitions supplied as command line arguments. These override definitions in the makefile and the environment both, but only for the make command itself.

make exports environment variables, using the most recently defined value. Macro definitions supplied on the command line are not normally exported, unless the macro is also an environment variable.

make does not export macros defined in the makefile. If an environment variable is set, and a macro with the same name is defined on the command line, make exports its value as defined on the command line. Unless -e is in effect, macro definitions within the makefile take precedence over those imported from the environment.

The macros MAKEFLAGS, MAKE, SHELL, HOST_ARCH, HOST_MACH, and TARGET_MACH are special cases. See Special-Purpose Macros below for details.

A target entry has the following format:

```
```

[command]

. . .
make(1S)

The first line contains the name of a target, or a space-separated list of target names, terminated with a colon or double colon. If a list of targets is given, this is equivalent to having a separate entry of the same form for each target. The colon(s) may be followed by a dependency, or a dependency list. make checks this list before building the target. The dependency list may be terminated with a semicolon (;), which in turn can be followed by a single Bourne shell command. Subsequent lines in the target entry begin with a TAB and contain Bourne shell commands. These commands comprise the rule for building the target.

Shell commands may be continued across input lines by escaping the NEWLINE with a backslash (\). The continuing line must also start with a TAB.

To rebuild a target, make expands macros, strips off initial TAB characters and either executes the command directly (if it contains no shell metacharacters), or passes each command line to a Bourne shell for execution.

The first non-empty line that does not begin with a TAB or ‘#’ begins another target or macro definition.

Global

Start a comment. The comment ends at the next NEWLINE. If the ‘#’ follows the TAB in a command line, that line is passed to the shell (which also treats ‘#’ as the start of a comment).

include filename

If the word include appears as the first seven letters of a line and is followed by a SPACE or TAB, the string that follows is taken as a filename to interpolate at that line. include files can be nested to a depth of no more than about 16. If filename is a macro reference, it is expanded.

Targets and Dependencies

Target list terminator. Words following the colon are added to the dependency list for the target or targets. If a target is named in more than one colon-terminated target entry, the dependencies for all its entries are added to form that target’s complete dependency list.

Target terminator for alternate dependencies. When used in place of a ‘:’ the double-colon allows a target to be checked and updated with respect to alternate dependency lists. When the target is out-of-date with respect to dependencies listed in the first alternate, it is built according to the rule for that entry. When out-of-date with respect to dependencies in another alternate, it is built according the rule in that other entry. Implicit rules do not apply to double-colon targets; you must supply a rule for each entry. If no dependencies are specified, the rule is always performed.
**Target group.** The rule in the target entry builds all the indicated targets as a group. It is normally performed only once per `make` run, but is checked for command dependencies every time a target in the group is encountered in the dependency scan.

`%` Pattern matching wild card metacharacter. Like the `*` shell wild card, `%` matches any string of zero or more characters in a target name or dependency, in the target portion of a conditional macro definition, or within a pattern replacement macro reference. Notice that only one `%` can appear in a target, dependency-name, or pattern-replacement macro reference.

`./pathname` `make` ignores the leading `./` characters from targets with names given as pathnames relative to "dot," the working directory.

**Macros**

= Macro definition. The word to the left of this character is the macro name; words to the right comprise its value. Leading and trailing white space characters are stripped from the value. A word break following the = is implied.

$ Macro reference. The following character, or the parenthesized or bracketed string, is interpreted as a macro reference: `make` expands the reference (including the $) by replacing it with the macro's value.

( ) ( ) Macro-reference name delimiters. A parenthesized or bracketed word appended to a $ is taken as the name of the macro being referred to. Without the delimiters, `make` recognizes only the first character as the macro name.

$$ A reference to the dollar-sign macro, the value of which is the character `$`. Used to pass variable expressions beginning with $ to the shell, to refer to environment variables which are expanded by the shell, or to delay processing of dynamic macros within the dependency list of a target, until that target is actually processed.

\$ Escaped dollar-sign character. Interpreted as a literal dollar sign within a rule.

+= When used in place of `=` , appends a string to a macro definition (must be surrounded by white space, unlike `=`).

:= Conditional macro assignment. When preceded by a list of targets with explicit target entries, the macro definition that follows takes effect when processing only those targets, and their dependencies.

:sh = Define the value of a macro to be the output of a command (see Command Substitutions below).
In a macro reference, execute the command stored in the macro, and replace the reference with the output of that command (see Command Substitutions below).

**Rules**

- `+` make will always execute the commands preceded by a “+”, even when `-n` is specified.
- `-` make ignores any nonzero error code returned by a command line for which the first non-TAB character is a ‘-’. This character is not passed to the shell as part of the command line. make normally terminates when a command returns nonzero status, unless the `-i` or `-k` options, or the .IGNORE: special-function target is in effect.
- `@` If the first non-TAB character is a @, make does not print the command line before executing it. This character is not passed to the shell.
- `?` Escape command-dependency checking. Command lines starting with this character are not subject to command dependency checking.
- `!` Force command-dependency checking. Command-dependency checking is applied to command lines for which it would otherwise be suppressed. This checking is normally suppressed for lines that contain references to the ‘?’ dynamic macro (for example, ‘$?’).

When any combination of ‘+’, ‘-’, ‘@’, ‘?’, or ‘!’ appear as the first characters after the TAB, all that are present apply. None are passed to the shell.

**Special-Function Targets**

When incorporated in a makefile, the following target names perform special-functions:

- `.DEFAULT:` If it has an entry in the makefile, the rule for this target is used to process a target when there is no other entry for it, no rule for building it, and no SCCS history file from which to retrieve a current version. make ignores any dependencies for this target.

- `.DONE:` If defined in the makefile, make processes this target and its dependencies after all other targets are built. This target is also performed when make halts with an error, unless the .FAILED target is defined.

- `.FAILED:` This target, along with its dependencies, is performed instead of .DONE when defined in the makefile and make halts with an error.
This target contains the rule for retrieving the current version of an SCCS file from its history file in the current working directory. `make` uses this rule when it is running in POSIX mode.

Ignore errors. When this target appears in the makefile, `make` ignores non-zero error codes returned from commands. When used in POSIX mode, `.IGNORE` could be followed by target names only, for which the errors will be ignored.

If defined in the makefile, this target and its dependencies are built before any other targets are processed.

If this target is in effect, `make` updates the state file, `.make.state`, in the current directory. This target also activates command dependencies, and hidden dependency checks. If either the `.KEEP_STATE:` target appears in the makefile, or the environment variable `KEEP_STATE` is set ("setenv KEEP_STATE"), `make` will rebuild everything in order to collect dependency information, even if all the targets were up to date due to previous `make` runs. See also the ENVIRONMENT VARIABLES section. This target has no effect if used in POSIX mode.

This target has no effect if used in POSIX mode. This target implies `.KEEP_STATE`. If the target is followed by a filename, `make` uses it as the state file. If the target is followed by a directory name, `make` looks for a `.make.state` file in that directory. If the target is not followed by any name, `make` looks for `.make.state` file in the current working directory.

A target-entry of the form:

```
ZMAKE_VERSION: VERSION-number
```

enables version checking. If the version of `make` differs from the version indicated, `make` issues a warning message.

Currently, this target has no effect, it is, however, reserved for future use.

Currently of no effect, but reserved for future use.

This target enables POSIX mode.

List of files not to delete. `make` does not remove any of the files listed as dependencies for this target when
interrupted. make normally removes the current target when it receives an interrupt. When used in POSIX mode, if the target is not followed by a list of files, all the file are assumed precious.

**.SCCS_GET:**
This target contains the rule for retrieving the current version of an SCCS file from its history file. To suppress automatic retrieval, add an entry for this target with an empty rule to your makefile.

**.SCCS_GET_POSIX:**
This target contains the rule for retrieving the current version of an SCCS file from its history file. make uses this rule when it is running in POSIX mode.

**.SILENT:**
Run silently. When this target appears in the makefile, make does not echo commands before executing them. When used in POSIX mode, it could be followed by target names, and only those will be executed silently.

**.SUFFIXES:**
The suffixes list for selecting implicit rules (see The Suffixes List).

**.WAIT:**
Currently of no effect, but reserved for future use.

### Clearing Special Targets
In this version of make, you can clear the definition of the following special targets by supplying entries for them with no dependencies and no rule:

```
.DEFAULT, .SCCS_GET, and .SUFFIXES
```

### Command Dependencies
When the .KEEP_STATE: target is effective, make checks the command for building a target against the state file. If the command has changed since the last make run, make rebuilds the target.

### Hidden Dependencies
When the .KEEP_STATE: target is effective, make reads reports from cpp(1) and other compilation processors for any “hidden” files, such as #include files. If the target is out of date with respect to any of these files, make rebuilds it.

### Macros
Entries of the form

```
macro=value
```

define macros. *macro* is the name of the macro, and *value*, which consists of all characters up to a comment character or unescaped NEWLINE, is the value. make strips both leading and trailing white space in accepting the value.

Subsequent references to the macro, of the forms: \$ (name) or \$ {name} are replaced by value. The parentheses or brackets can be omitted in a reference to a macro with a single-character name.

Macro references can contain references to other macros, in which case nested references are expanded first.
Substitutions within macros can be made as follows:

$\langle \text{name} : \text{string1} = \text{string2} \rangle$

where \text{string1} is either a suffix, or a word to be replaced in the macro definition, and \text{string2} is the replacement suffix or word. Words in a macro value are separated by SPACE, TAB, and escaped NEWLINE characters.

Pattern matching replacements can also be applied to macros, with a reference of the form:

\$\langle \text{name} : \text{op}\%\text{os} = \text{np}\%\text{ns} \rangle$

where \text{op} is the existing (old) prefix and \text{os} is the existing (old) suffix, \text{np} and \text{ns} are the new prefix and new suffix, respectively, and the pattern matched by \% (a string of zero or more characters), is carried forward from the value being replaced. For example:

\begin{verbatim}
PROGRAM=fabricate
DEBUG= $(PROGRAM:%=tmp/%−g)
\end{verbatim}

Notice that pattern replacement macro references cannot be used in the dependency list of a pattern matching rule; the \% characters are not evaluated independently. Also, any number of \% metacharacters can appear after the equal-sign.

Words can be appended to macro values as follows:

\begin{verbatim}
macro += word...
\end{verbatim}

When the \text{MAKEFLAGS} variable is present in the environment, \text{make} takes options from it, in combination with options entered on the command line. \text{make} retains this combined value as the \text{MAKEFLAGS} macro, and exports it automatically to each command or shell it invokes.

Notice that flags passed by way of \text{MAKEFLAGS} are only displayed when the -d, or -dd options are in effect.

The \text{MAKE} macro is another special case. It has the value \text{make} by default, and temporarily overrides the -n option for any line in which it is referred to. This allows nested invocations of \text{make} written as:

\begin{verbatim}
$\langle \text{MAKE} \rangle \ldots$
\end{verbatim}

to run recursively, with the -n flag in effect for all commands but \text{make}. This lets you use ‘\text{make} -n’ to test an entire hierarchy of \text{make} files.

For compatibility with the 4.2 BSD \text{make}, the \text{MFLAGS} macro is set from the \text{MAKEFLAGS} variable by prepending a ‘-’. \text{MFLAGS} is not exported automatically.

The \text{SHELL} macro, when set to a single-word value such as /\text{usr}/\text{bin}/csh, indicates the name of an alternate shell to use. The default is /\text{bin}/sh. Notice that \text{make} executes commands that contain no shell metacharacters itself. Built-in commands, such as \text{dirs} in the C shell, are not recognized unless the command line includes a
metacharacter (for instance, a semicolon). This macro is neither imported from, nor exported to the environment, regardless of -e. To be sure it is set properly, you must define this macro within every makefile that requires it.

The syntax of the VPATH macro is:

```plaintext
VPATH = [ pathname [ : pathname ] ... ]
```

VPATH specifies a list of directories to search for the files, which are targets or dependencies, when make is executed. VPATH is also used in order to search for the include files mentioned in the particular makefile.

When processing a target or a dependency or an include directive, make checks the existence of the file with the same name in the current directory. If the file is found to be missing, make will search for this file in the list of directories presented in VPATH (like the PATH variable in the shell). Unlike the PATH variable, VPATH is used in order to search for the files with relative pathnames. When make attempts to apply implicit rules to the target, it also searches for the dependency files using VPATH.

When the file is found using VPATH, internal macros $@, @<, $?,$*, and their alternative forms (with D or F appended) are set in accordance with the name derived from VPATH. For instance, if the target subdir/foo.o is found in the directory /aaa/bbb using VPATH, then the value of the internal macro $@ for this target will be /aaa/bbb/subdir/foo.o.

If a target or a dependency file is found using VPATH, then any occurrences of the word that is the same as the target name in the subsequent rules will be replaced with the actual name of the target derived from VPATH.

For example:

```plaintext
VPATH=./subdir
file.o : file.c
    cc -c file.c -o file.o
```

If file.c is found in ./subdir, then the command

```plaintext
cc -c ./subdir/file.c -o file.o
```

will be executed.

The following macros are provided for use with cross-compilation:

- **HOST_ARCH**: The machine architecture of the host system. By default, this is the output of the arch(1) command prepended with ‘–’. Under normal circumstances, this value should never be altered by the user.

- **HOST_MACH**: The machine architecture of the host system. By default, this is the output of the mach(1), prepended with ‘–’. Under normal circumstances, this value should never be altered by the user.

- **TARGET_ARCH**: The machine architecture of the target system. By default, the output of mach, prepended with ‘–’.
There are several dynamically maintained macros that are useful as abbreviations within rules. They are shown here as references; if you were to define them, make would simply override the definition.

$*$  The basename of the current target, derived as if selected for use with an implicit rule.

$<$  The name of a dependency file, derived as if selected for use with an implicit rule.

$@$  The name of the current target. This is the only dynamic macro whose value is strictly determined when used in a dependency list. (In which case it takes the form ‘$$@’.)

$?$  The list of dependencies that are newer than the target. Command-dependency checking is automatically suppressed for lines that contain this macro, just as if the command had been prefixed with a ‘?’ . See the description of ‘? ’, under Special Character Rules above. You can force this check with the ! command-line prefix.

$%  The name of the library member being processed. (See Library Maintenance below.)

To refer to the $@$ dynamic macro within a dependency list, precede the reference with an additional ‘$’ character (as in, ‘$$@’). Because make assigns $<$ and $*$ as it would for implicit rules (according to the suffixes list and the directory contents), they may be unreliable when used within explicit target entries.

These macros can be modified to apply either to the filename part, or the directory part of the strings they stand for, by adding an upper case F or D, respectively (and enclosing the resulting name in parentheses or braces). Thus, ‘$(@D)’ refers to the directory part of the string ‘$@’; if there is no directory part, ‘.’ is assigned. $(@F) refers to the filename part.

Conditional Macro Definitions

A macro definition of the form:

```
target-list := macro = value
```

indicates that when processing any of the targets listed and their dependencies, macro is to be set to the value supplied. Notice that if a conditional macro is referred to in a dependency list, the $ must be delayed (use $$ instead). Also, target-list may contain a % pattern, in which case the macro will be conditionally defined for all targets encountered that match the pattern. A pattern replacement reference can be used within the value.

You can temporarily append to a macro’s value with a conditional definition of the form:

```
target-list := macro += value
```

Predefined Macros

make supplies the macros shown in the table that follows for compilers and their options, host architectures, and other commands. Unless these macros are read in as environment variables, their values are not exported by make. If you run make with
any of these set in the environment, it is a good idea to add commentary to the makefile to indicate what value each is expected to take. If -r is in effect, make does not read the default makefile (.make.rules or /usr/share/lib/make/make.rules) in which these macro definitions are supplied.

<table>
<thead>
<tr>
<th>Use</th>
<th>Macro</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Library</td>
<td>AR</td>
<td>ar</td>
</tr>
<tr>
<td></td>
<td>ARFLAGS</td>
<td>rv</td>
</tr>
<tr>
<td>Archives</td>
<td>AS</td>
<td>as</td>
</tr>
<tr>
<td></td>
<td>ASFLAGS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>COMPILE.s</td>
<td>$(AS) $(ASFLAGS)</td>
</tr>
<tr>
<td></td>
<td>COMPILE.S</td>
<td>$(CC) $(ASFLAGS) $(CPPFLAGS) -c</td>
</tr>
<tr>
<td>Assembler</td>
<td>C</td>
<td>cc</td>
</tr>
<tr>
<td>Commands</td>
<td>CC</td>
<td>cc</td>
</tr>
<tr>
<td></td>
<td>CFLAGS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPPFLAGS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>COMPILE.c</td>
<td>$(CC) $(CFLAGS) $(CPPFLAGS) -c</td>
</tr>
<tr>
<td></td>
<td>LINK.c</td>
<td>$(CC) $(CFLAGS) $(CPPFLAGS) $(LDFLAGS)</td>
</tr>
<tr>
<td></td>
<td>CCC</td>
<td>cc</td>
</tr>
<tr>
<td>Compiler</td>
<td>CC</td>
<td>cc</td>
</tr>
<tr>
<td></td>
<td>CCFLAGS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPPFLAGS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>COMPILE.cc</td>
<td>$(CCC) $(CCFLAGS) $(CPPFLAGS) -c</td>
</tr>
<tr>
<td></td>
<td>LINK.cc</td>
<td>$(CCC) $(CCFLAGS) $(CPPFLAGS) $(LDFLAGS)</td>
</tr>
<tr>
<td></td>
<td>Compile</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CC</td>
<td>cc</td>
</tr>
<tr>
<td></td>
<td>CCFLAGS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPPFLAGS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>COMPILE.cc</td>
<td>$(CCC) $(CCFLAGS) $(CPPFLAGS) -c</td>
</tr>
<tr>
<td></td>
<td>LINK.cc</td>
<td>$(CCC) $(CCFLAGS) $(CPPFLAGS) $(LDFLAGS)</td>
</tr>
<tr>
<td></td>
<td>FORTRAN 77</td>
<td>f77</td>
</tr>
<tr>
<td></td>
<td>FFLAGS</td>
<td></td>
</tr>
</tbody>
</table>

Table of Predefined Macros
Table of Predefined Macros

<table>
<thead>
<tr>
<th>Use</th>
<th>Macro</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commands</td>
<td>COMPILE.f</td>
<td>$(FC) $(FFLAGS) -c</td>
</tr>
<tr>
<td></td>
<td>LINK.f</td>
<td>$(FC) $(FFLAGS) $(LDFLAGS)</td>
</tr>
<tr>
<td></td>
<td>COMPILE.F</td>
<td>$(FC) $(FFLAGS) $(CPPFLAGS) -c</td>
</tr>
<tr>
<td></td>
<td>LINK.F</td>
<td>$(FC) $(FFLAGS) $(CPPFLAGS) $(LDFLAGS)</td>
</tr>
<tr>
<td></td>
<td>FORTRAN 90</td>
<td>FC f90</td>
</tr>
<tr>
<td>Compiler</td>
<td>F90FLAGS</td>
<td></td>
</tr>
<tr>
<td>Commands</td>
<td>COMPILE.f90</td>
<td>$(F90C) $(F90FLAGS) -c</td>
</tr>
<tr>
<td></td>
<td>LINK.f90</td>
<td>$(F90C) $(F90FLAGS) $(LDFLAGS)</td>
</tr>
<tr>
<td></td>
<td>COMPILE.ftn</td>
<td>$(F90C) $(F90FLAGS) $(CPPFLAGS) -c</td>
</tr>
<tr>
<td></td>
<td>LINK.ftn</td>
<td>$(F90C) $(F90FLAGS) $(CPPFLAGS) $(LDFLAGS)</td>
</tr>
<tr>
<td>Link Editor</td>
<td>LD</td>
<td>ld</td>
</tr>
<tr>
<td>Command</td>
<td>LDFLAGS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>lex</td>
<td>LEX lex</td>
</tr>
<tr>
<td>Command</td>
<td>LFLAGS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LEX.l</td>
<td>$(LEX) $(LFLAGS) -t</td>
</tr>
<tr>
<td></td>
<td>lint</td>
<td>LINT lint</td>
</tr>
<tr>
<td>Command</td>
<td>LINTFLAGS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LINT.c</td>
<td>$(LINT) $(LINTFLAGS) $(CPPFLAGS)</td>
</tr>
<tr>
<td></td>
<td>Modula 2</td>
<td>M2C m2c</td>
</tr>
<tr>
<td>Commands</td>
<td>M2FLAGS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MODFLAGS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DEFFLAGS</td>
<td></td>
</tr>
</tbody>
</table>

User Commands 831
### Table of Predefined Macros

<table>
<thead>
<tr>
<th>Use</th>
<th>Macro</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>COMPILE.def</td>
<td>$(M2C) $(M2FLAGS) $(DEFFLAGS)</td>
</tr>
<tr>
<td></td>
<td>COMPILE.mod</td>
<td>$(M2C) $(M2FLAGS) $(MODFLAGS)</td>
</tr>
<tr>
<td>Pascal Compiler</td>
<td>PC</td>
<td>pc</td>
</tr>
<tr>
<td></td>
<td>PFLAGS</td>
<td></td>
</tr>
<tr>
<td>Commands</td>
<td>COMPILE.p</td>
<td>$(PC) $(PFLAGS) $(CPPFLAGS) -c</td>
</tr>
<tr>
<td></td>
<td>LINK.p</td>
<td>$(PC) $(PFLAGS) $(CPPFLAGS) $(LDFLAGS)</td>
</tr>
<tr>
<td>Ratfor Compilation</td>
<td>COMPILE.r</td>
<td>$(PC) $(PFLAGS) $(RFLAGS) -c</td>
</tr>
<tr>
<td></td>
<td>LINK.r</td>
<td>$(PC) $(PFLAGS) $(RFLAGS) $(LDFLAGS)</td>
</tr>
<tr>
<td>rm Command</td>
<td>RM</td>
<td>rm-f</td>
</tr>
<tr>
<td>sccs Command</td>
<td>SCCSFLAGS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SCCSGETFLAGS</td>
<td>-s</td>
</tr>
<tr>
<td>yacc Command</td>
<td>YACC</td>
<td>yacc</td>
</tr>
<tr>
<td></td>
<td>YFLAGS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>YACC.y</td>
<td>$(YACC) $(YFLAGS)</td>
</tr>
</tbody>
</table>

### Implicit Rules

When a target has no entry in the makefile, `make` attempts to determine its class (if any) and apply the rule for that class. An implicit rule describes how to build any target of a given class, from an associated dependency file. The class of a target can be
make(1S)

Pattern Matching Rules

determined either by a pattern, or by a suffix; the corresponding dependency file (with the same basename) from which such a target might be built. In addition to a predefined set of implicit rules, make allows you to define your own, either by pattern, or by suffix.

A target entry of the form:

\texttt{tp\%ts: dp\%ds rule}

is a pattern matching rule, in which \texttt{tp} is a target prefix, \texttt{ts} is a target suffix, \texttt{dp} is a dependency prefix, and \texttt{ds} is a dependency suffix (any of which may be null). The \texttt{\%} stands for a basename of zero or more characters that is matched in the target, and is used to construct the name of a dependency. When make encounters a match in its search for an implicit rule, it uses the rule in that target entry to build the target from the dependency file. Pattern-matching implicit rules typically make use of the \texttt{$@} and \texttt{$<$} dynamic macros as placeholders for the target and dependency names. Other, regular dependencies may occur in the dependency list; however, none of the regular dependencies may contain \texttt{\%}. An entry of the form:

\texttt{tp\%ts: \{dependency... \} dp\%ds \{dependency... \} rule}

is a valid pattern matching rule.

Suffix Rules

When no pattern matching rule applies, make checks the target name to see if it ends with a suffix in the known suffixes list. If so, make checks for any suffix rules, as well as a dependency file with same root and another recognized suffix, from which to build it.

The target entry for a suffix rule takes the form:

\texttt{DsTs: rule}

where \texttt{Ts} is the suffix of the target, \texttt{Ds} is the suffix of the dependency file, and \texttt{rule} is the rule for building a target in the class. Both \texttt{Ds} and \texttt{Ts} must appear in the suffixes list. (A suffix need not begin with a \texttt{.} to be recognized.)

A suffix rule with only one suffix describes how to build a target having a null (or no) suffix from a dependency file with the indicated suffix. For instance, the \texttt{.c} rule could be used to build an executable program named \texttt{file} from a C source file named \texttt{file.c}. If a target with a null suffix has an explicit dependency, make omits the search for a suffix rule.
## Table of Standard Implicit (Suffix) Rules for Assembly Files

<table>
<thead>
<tr>
<th>Implicit Rule Name</th>
<th>Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>.s.o</code></td>
<td><code>$(COMPILE.s) -o $@ $&lt;</code></td>
</tr>
<tr>
<td><code>.s.a</code></td>
<td><code>$(COMPILE.s) -o $% $&lt;</code></td>
</tr>
<tr>
<td></td>
<td><code>$(AR) $(ARFLAGS) $@ $%</code></td>
</tr>
<tr>
<td></td>
<td><code>$(RM) $%</code></td>
</tr>
<tr>
<td><code>.s~.o</code></td>
<td><code>$(GET) $(GFLAGS) -p $&lt; &gt; $*.s</code></td>
</tr>
<tr>
<td></td>
<td><code>$(COMPILE.s) -o $@ $*.s</code></td>
</tr>
<tr>
<td><code>.S.o</code></td>
<td><code>$(COMPILE.S) -o $@ $&lt;</code></td>
</tr>
<tr>
<td><code>.S.a</code></td>
<td><code>$(COMPILE.S) -o $% $&lt;</code></td>
</tr>
<tr>
<td></td>
<td><code>$(AR) $(ARFLAGS) $@ $%</code></td>
</tr>
<tr>
<td></td>
<td><code>$(RM) $%</code></td>
</tr>
<tr>
<td><code>.S~.o</code></td>
<td><code>$(GET) $(GFLAGS) -p $&lt; &gt; $*.S</code></td>
</tr>
<tr>
<td></td>
<td><code>$(COMPILE.S) -o $@ $*.S</code></td>
</tr>
<tr>
<td><code>.S~.a</code></td>
<td><code>$(GET) $(GFLAGS) -p $&lt; &gt; $*.S</code></td>
</tr>
<tr>
<td></td>
<td><code>$(COMPILE.S) -o $% $*.S</code></td>
</tr>
<tr>
<td></td>
<td><code>$(AR) $(ARFLAGS) $@ $%</code></td>
</tr>
<tr>
<td></td>
<td><code>$(RM) $%</code></td>
</tr>
</tbody>
</table>

## Table of Standard Implicit (Suffix) Rules for C Files

<table>
<thead>
<tr>
<th>Implicit Rule Name</th>
<th>Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>.c</code></td>
<td><code>$(LINK.c) -o $@ $&lt; $(LDLIBS)</code></td>
</tr>
<tr>
<td><code>.c.ln</code></td>
<td><code>$(LINT.c) $(OUTPUT_OPTION) -i $&lt;</code></td>
</tr>
</tbody>
</table>
### Table of Standard Implicit (Suffix) Rules for C Files

<table>
<thead>
<tr>
<th>Implicit Rule Name</th>
<th>Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>.c.o</td>
<td>$(COMPILE.c) $(OUTPUT_OPTION) $&lt;</td>
</tr>
<tr>
<td>.c.a</td>
<td>$(COMPILE.c) -o $% $&lt;</td>
</tr>
<tr>
<td></td>
<td>$(AR) $(ARFLAGS) $@ $%</td>
</tr>
<tr>
<td></td>
<td>$(RM) $%</td>
</tr>
<tr>
<td>.c-</td>
<td>$(GET) $(GFLAGS) -p $&lt; &gt; $*.c</td>
</tr>
<tr>
<td></td>
<td>$(CC) $(CFLAGS) $(LDFLAGS) -o $@ $*.c</td>
</tr>
<tr>
<td>.c-.o</td>
<td>$(GET) $(GFLAGS) -p $&lt; &gt; $*.c</td>
</tr>
<tr>
<td></td>
<td>$(CC) $(CFLAGS) -c $*.c</td>
</tr>
<tr>
<td>.c-.ln</td>
<td>$(GET) $(GFLAGS) -p $&lt; &gt; $*.c</td>
</tr>
<tr>
<td></td>
<td>$(LINT.c) $(OUTPUT_OPTION) -c $*.c</td>
</tr>
<tr>
<td>.c-.a</td>
<td>$(GET) $(GFLAGS) -p $&lt; &gt; $*.c</td>
</tr>
<tr>
<td></td>
<td>$(COMPILE.c) -o $% $*.c</td>
</tr>
<tr>
<td></td>
<td>$(AR) $(ARFLAGS) $@ $%</td>
</tr>
<tr>
<td></td>
<td>$(RM) $%</td>
</tr>
</tbody>
</table>

### Table of Standard Implicit (Suffix) Rules for C++ Files

<table>
<thead>
<tr>
<th>Implicit Rule Name</th>
<th>Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>.cc</td>
<td>$(LINK.cc) -o $@ $&lt; $(LDBLIBS)</td>
</tr>
<tr>
<td>.cc.o</td>
<td>$(COMPILE.cc) $(OUTPUT_OPTION) $&lt;</td>
</tr>
<tr>
<td>.cc.a</td>
<td>$(COMPILE.cc) -o $% $&lt;</td>
</tr>
</tbody>
</table>
### Table of Standard Implicit (Suffix) Rules for C++ Files

<table>
<thead>
<tr>
<th>Implicit Rule Name</th>
<th>Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(AR) $(ARFLAGS) $@ $%</td>
<td>$(AR) $(ARFLAGS) $@ $%</td>
</tr>
<tr>
<td>$ (RM) $%</td>
<td>$ (RM) $%</td>
</tr>
<tr>
<td>$.cc~</td>
<td>$(GET) $(GFLAGS) -p $&lt; &gt; $*.cc</td>
</tr>
<tr>
<td>$ (LINK.cc) -o $@ $*.cc $(LDLIBS)</td>
<td>$ (LINK.cc) -o $@ $*.cc $(LDLIBS)</td>
</tr>
<tr>
<td>$.cc.o</td>
<td>$(COMPILE.cc) $(OUTPUT_OPTION) $&lt;</td>
</tr>
<tr>
<td>$.cc-.o</td>
<td>$(GET) $(GFLAGS) -p $&lt; &gt; $*.cc</td>
</tr>
<tr>
<td>$ (COMPILE.cc) $(OUTPUT_OPTION) $*.cc</td>
<td>$ (COMPILE.cc) $(OUTPUT_OPTION) $*.cc</td>
</tr>
<tr>
<td>$.cc.a</td>
<td>$(COMPILE.cc) -o $% $&lt;</td>
</tr>
<tr>
<td>$ (AR) $(ARFLAGS) $@ $%</td>
<td>$ (AR) $(ARFLAGS) $@ $%</td>
</tr>
<tr>
<td>$ (RM) $%</td>
<td>$ (RM) $%</td>
</tr>
<tr>
<td>$.cc-.a</td>
<td>$(GET) $(GFLAGS) -p $&lt; &gt; $*.cc</td>
</tr>
<tr>
<td>$ (COMPILE.cc) -o $% $*.cc</td>
<td>$ (COMPILE.cc) -o $% $*.cc</td>
</tr>
<tr>
<td>$ (AR) $(ARFLAGS) $@ $%</td>
<td>$ (AR) $(ARFLAGS) $@ $%</td>
</tr>
<tr>
<td>$ (RM) $%</td>
<td>$ (RM) $%</td>
</tr>
<tr>
<td>$.C</td>
<td>$(LINK.C) -o $@ $&lt; $(LDLIBS)</td>
</tr>
<tr>
<td>$.C~</td>
<td>$(GET) $(GFLAGS) -p $&lt; &gt; $*.C</td>
</tr>
<tr>
<td>$ (LINK.C) -o $@ $*.C $(LDLIBS)</td>
<td>$ (LINK.C) -o $@ $*.C $(LDLIBS)</td>
</tr>
<tr>
<td>$.C.o</td>
<td>$(COMPILE.C) $(OUTPUT_OPTION) $&lt;</td>
</tr>
<tr>
<td>$.C-.o</td>
<td>$(GET) $(GFLAGS) -p $&lt; &gt; $*.C</td>
</tr>
</tbody>
</table>
### Table of Standard Implicit (Suffix) Rules for C++ Files

<table>
<thead>
<tr>
<th>Implicit Rule Name</th>
<th>Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(COMPILE.C) $(OUTPUT_OPTION) $*.C</td>
<td>$(COMPILE.C) -o $% $&lt; $(AR) $(ARFLAGS) $@ $% $(RM) $%</td>
</tr>
</tbody>
</table>

### Table of Standard Implicit (Suffix) Rules for FORTRAN 77 Files

<table>
<thead>
<tr>
<th>Implicit Rule Name</th>
<th>Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>.f</td>
<td>$(LINK.f) -o $@ $&lt; $(LDLIBS)</td>
</tr>
<tr>
<td>.f.o</td>
<td>$(COMPILE.f) $(OUTPUT_OPTION) $&lt;</td>
</tr>
<tr>
<td>.f.a</td>
<td>$(COMPILE.f) -o $% $&lt; $(AR) $(ARFLAGS) $@ $% $(RM) $%</td>
</tr>
<tr>
<td>.f</td>
<td>$(LINK.f) -o $@ $&lt; $(LDLIBS)</td>
</tr>
<tr>
<td>.f~</td>
<td>$(GET) $(GFLAGS) -p $% $&lt; $<em>.f $(FC) $(FFLAGS) $(LDFLAGS) -o $@ $</em>.f</td>
</tr>
<tr>
<td>.f~.o</td>
<td>$(GET) $(GFLAGS) -p $% $&lt; $<em>.f $(FC) $(FFLAGS) -c $</em>.f</td>
</tr>
</tbody>
</table>
### Table of Standard Implicit (Suffix) Rules for FORTRAN 77 Files

<table>
<thead>
<tr>
<th>Implicit Rule Name</th>
<th>Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>.f~.a</code></td>
<td><code>$(GET) $(GFLAGS) -p $&lt; &gt; $*.f</code>&lt;br&gt;<code>$(COMPILE.f) -o $% $*.f</code>&lt;br&gt;<code>$(AR) $(ARFLAGS) $@ $%</code>&lt;br&gt;<code>$(RM) $%</code></td>
</tr>
<tr>
<td><code>.F</code></td>
<td><code>$(LINK.F) -o $@ $&lt; $(LDLIBS)</code></td>
</tr>
<tr>
<td><code>.F.o</code></td>
<td><code>$(COMPILE.F) $(OUTPUT_OPTION) $&lt;</code></td>
</tr>
<tr>
<td><code>.F.a</code></td>
<td><code>$(COMPILE.F) -o $% $&lt;</code>&lt;br&gt;<code>$(AR) $(ARFLAGS) $@ $%</code>&lt;br&gt;<code>$(RM) $%</code></td>
</tr>
<tr>
<td><code>.F~</code></td>
<td><code>$(GET) $(GFLAGS) -p $&lt; &gt; $*.F</code>&lt;br&gt;<code>$(FC) $(FFLAGS) $(LDFLAGS) -o $@ $*.F</code></td>
</tr>
<tr>
<td><code>.F~.o</code></td>
<td><code>$(GET) $(GFLAGS) -p $&lt; &gt; $*.F</code>&lt;br&gt;<code>$(FC) $(FFLAGS) -o $*.F</code></td>
</tr>
<tr>
<td><code>.F~.a</code></td>
<td><code>$(GET) $(GFLAGS) -p $&lt; &gt; $*.F</code>&lt;br&gt;<code>$(COMPILE.F) -o $% $*.F</code>&lt;br&gt;<code>$(AR) $(ARFLAGS) $@ $%</code>&lt;br&gt;<code>$(RM) $%</code></td>
</tr>
</tbody>
</table>

### Table of Standard Implicit (Suffix) Rules for FORTRAN 90 Files

<table>
<thead>
<tr>
<th>Implicit Rule Name</th>
<th>Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>.f90</code></td>
<td><code>$(LINK.f90) -o $@ $&lt; $(LDLIBS)</code></td>
</tr>
<tr>
<td>Implicit Rule Name</td>
<td>Command Line</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>.f90~</code></td>
<td><code>$(GET) $(GFLAGS) -p $&lt; &gt; $*.f90 $(LINK.f90) -o $@ $*.f90 $(LDLIBS)</code></td>
</tr>
<tr>
<td><code>.f90.o</code></td>
<td><code>$(COMPILE.f90) $(OUTPUT_OPTION) $&lt;</code></td>
</tr>
<tr>
<td><code>.f90-.o</code></td>
<td><code>$(GET) $(GFLAGS) -p $&lt; &gt; $*.f90 $(COMPILE.f90) $(OUTPUT_OPTION) $*.f90</code></td>
</tr>
<tr>
<td><code>.f90.a</code></td>
<td><code>$(COMPILE.f90) -o % $&lt; $(AR) $(ARFLAGS) $@ % $(RM) %</code></td>
</tr>
<tr>
<td><code>.f90-.a</code></td>
<td><code>$(GET) $(GFLAGS) -p $&lt; &gt; $*.f90 $(COMPILE.f90) -o % $*.f90 $(AR) $(ARFLAGS) $@ % $(RM) %</code></td>
</tr>
<tr>
<td><code>.ftn</code></td>
<td><code>$(LINK.ftn) -o $@ $&lt; $(LDLIBS)</code></td>
</tr>
<tr>
<td><code>.ftn~</code></td>
<td><code>$(GET) $(GFLAGS) -p $&lt; &gt; $*.ftn $(LINK.ftn) -o $@ $*.ftn $(LDLIBS)</code></td>
</tr>
<tr>
<td><code>.ftn.o</code></td>
<td><code>$(COMPILE.ftn) $(OUTPUT_OPTION) $&lt;</code></td>
</tr>
<tr>
<td><code>.ftn-.o</code></td>
<td><code>$(GET) $(GFLAGS) -p $&lt; &gt; $*.ftn $(COMPILE.ftn) $(OUTPUT_OPTION) $*.ftn</code></td>
</tr>
</tbody>
</table>
Table of Standard Implicit (Suffix) Rules for FORTRAN 90 Files

<table>
<thead>
<tr>
<th>Implicit Rule Name</th>
<th>Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>.ftn.a</td>
<td>$(COMPILE.ftn) -o % $&lt;&lt;br&gt;$ (AR) $(ARFLAGS) $@ %&lt;br&gt;$ (RM) %</td>
</tr>
<tr>
<td>.ftn-.a</td>
<td>$(GET) $(GFLAGS) -p $&lt; &gt; $<em>.ftn&lt;br&gt;$ (COMPILE.ftn) -o % $</em>.ftn&lt;br&gt;$ (AR) $(ARFLAGS) $@ %&lt;br&gt;$ (RM) %</td>
</tr>
</tbody>
</table>

Table of Standard Implicit (Suffix) Rules for lex Files

<table>
<thead>
<tr>
<th>Implicit Rule Name</th>
<th>Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>.l</td>
<td>$(RM) $<em>.c&lt;br&gt;$ (LEX.l) $&lt; &gt; $</em>.c&lt;br&gt;$ (LINK.c) -o $@ $<em>.c $(LDLIBS)&lt;br&gt;$ (RM) $</em>.c</td>
</tr>
<tr>
<td>.l.c</td>
<td>$(RM) $@&lt;br&gt;$ (LEX.l) $&lt; &gt; $@</td>
</tr>
<tr>
<td>.l.ln</td>
<td>$(RM) $<em>.c&lt;br&gt;$ (LEX.l) $&lt; &gt; $</em>.c&lt;br&gt;$ (LINT.c) -o $@ -i $<em>.c&lt;br&gt;$ (RM) $</em>.c</td>
</tr>
<tr>
<td>.l.o</td>
<td>$(RM) $<em>.c&lt;br&gt;$ (LEX.l) $&lt; &gt; $</em>.c&lt;br&gt;$ (COMPILE.c) -o $@ $<em>.c&lt;br&gt;$ (RM) $</em>.c</td>
</tr>
</tbody>
</table>
### Table of Standard Implicit (Suffix) Rules for lex Files

<table>
<thead>
<tr>
<th>Implicit Rule Name</th>
<th>Command Line</th>
</tr>
</thead>
</table>
| `.l~` | $(GET) $(GFLAGS) -p $< > $*.l  
$(LEX) $(LFLAGS) $*.l  
$(CC) $(CFLAGS) -c lex.yy.c  
rm -f lex.yy.c  
mv lex.yy.c $@ |
| `.l~.c` | $(GET) $(GFLAGS) -p $< > $*.l  
$(LEX) $(LFLAGS) $*.l  
mv lex.yy.c $@ |
| `.l~.ln` | $(GET) $(GFLAGS) -p $< > $*.l  
$(RM) $*.c  
$(LEX.l) $*.l > $*.c  
$(LINT.c) -o $@ -i $*.c  
$(RM) $*.c |
| `.l~.o` | $(GET) $(GFLAGS) -p $< > $*.l  
$(LEX) $(LFLAGS) $*.l  
$(CC) $(CFLAGS) -c lex.yy.c  
rm -f lex.yy.c  
mv lex.yy.c $@ |

### Table of Standard Implicit (Suffix) Rules for Modula 2 Files

<table>
<thead>
<tr>
<th>Implicit Rule Name</th>
<th>Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>.mod</code></td>
<td>$(COMPILE.mod) -o $@ -e $@ $&lt;</td>
</tr>
<tr>
<td><code>.mod.o</code></td>
<td>$(COMPILE.mod) -o $@ $&lt;</td>
</tr>
</tbody>
</table>
### Table of Standard Implicit (Suffix) Rules for Modula 2 Files

<table>
<thead>
<tr>
<th>Implicit Rule Name</th>
<th>Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>.def.sym</td>
<td><code>$(COMPILE.def) -o $@ $&lt;</code></td>
</tr>
<tr>
<td>.def~.sym</td>
<td><code>$(GET) $(GFLAGS) -p $&lt; &gt; $*.def</code></td>
</tr>
<tr>
<td></td>
<td><code>$(COMPILE.def) -o $@ $*.def</code></td>
</tr>
<tr>
<td>.mod~</td>
<td><code>$(GET) $(GFLAGS) -p $&lt; &gt; $*.mod</code></td>
</tr>
<tr>
<td></td>
<td><code>$(COMPILE.mod) -o $@ -e $@ $*.mod</code></td>
</tr>
<tr>
<td>.mod~.o</td>
<td><code>$(GET) $(GFLAGS) -p $&lt; &gt; $*.mod</code></td>
</tr>
<tr>
<td></td>
<td><code>$(COMPILE.mod) -o $@ $*.mod</code></td>
</tr>
<tr>
<td>.mod~.a</td>
<td><code>$(GET) $(GFLAGS) -p $&lt; &gt; $*.mod</code></td>
</tr>
<tr>
<td></td>
<td><code>$(COMPILE.mod) -o $% $*.mod</code></td>
</tr>
<tr>
<td></td>
<td><code>$(AR) $(ARFLAGS) $% $%</code></td>
</tr>
<tr>
<td></td>
<td><code>$(RM) $%</code></td>
</tr>
</tbody>
</table>

### Table of Standard Implicit (Suffix) Rules for NeWS Files

<table>
<thead>
<tr>
<th>Implicit Rule Name</th>
<th>Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>.cps.h</td>
<td><code>cps $*.cps</code></td>
</tr>
<tr>
<td>.cps~.h</td>
<td><code>$(GET) $(GFLAGS) -p $&lt; &gt; $*.cps</code></td>
</tr>
<tr>
<td></td>
<td><code>$(CPS) $(CPSFLAGS) $*.cps</code></td>
</tr>
</tbody>
</table>

### Table of Standard Implicit (Suffix) Rules for Pascal Files

<table>
<thead>
<tr>
<th>Implicit Rule Name</th>
<th>Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>.p</td>
<td><code>$(LINK.p) -o $@ $&lt; $(LDDLBS)</code></td>
</tr>
<tr>
<td>.p.o</td>
<td><code>$(COMPILE.p) $(OUTPUT_OPTION) $&lt;</code></td>
</tr>
</tbody>
</table>
### Table of Standard Implicit (Suffix) Rules for Pascal Files

<table>
<thead>
<tr>
<th>Implicit Rule Name</th>
<th>Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>.p~</td>
<td>$(GET) $(GFLAGS) -p $&lt; $*.p</td>
</tr>
<tr>
<td></td>
<td>$(LINK.p) -o $@ $*.p $(LDLIBS)</td>
</tr>
<tr>
<td>.p~.o</td>
<td>$(GET) $(GFLAGS) -p $&lt; $*.p</td>
</tr>
<tr>
<td></td>
<td>$(COMPILE.p) $(OUTPUT_OPTION) $*.p</td>
</tr>
<tr>
<td>.p~.a</td>
<td>$(GET) $(GFLAGS) -p $&lt; $*.p</td>
</tr>
<tr>
<td></td>
<td>$(COMPILE.p) -o $% $*.p</td>
</tr>
<tr>
<td></td>
<td>$(AR) $(ARFLAGS) $@ $%</td>
</tr>
<tr>
<td></td>
<td>$(RM) $%</td>
</tr>
</tbody>
</table>

### Table of Standard Implicit (Suffix) Rules for Ratfor Files

<table>
<thead>
<tr>
<th>Implicit Rule Name</th>
<th>Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>.r</td>
<td>$(LINK.r) -o $@ $&lt; $(LDLIBS)</td>
</tr>
<tr>
<td>.r.o</td>
<td>$(COMPILE.r) $(OUTPUT_OPTION) $&lt;</td>
</tr>
<tr>
<td>.r.a</td>
<td>$(COMPILE.r) -o $% $&lt;</td>
</tr>
<tr>
<td></td>
<td>$(AR) $(ARFLAGS) $@ $%</td>
</tr>
<tr>
<td></td>
<td>$(RM) $%</td>
</tr>
<tr>
<td>.r~</td>
<td>$(GET) $(GFLAGS) -p $&lt; $*.r</td>
</tr>
<tr>
<td></td>
<td>$(LINK.r) -o $@ $*.r $(LDLIBS)</td>
</tr>
<tr>
<td>.r~.o</td>
<td>$(GET) $(GFLAGS) -p $&lt; $*.r</td>
</tr>
<tr>
<td></td>
<td>$(COMPILE.r) $(OUTPUT_OPTION) $*.r</td>
</tr>
</tbody>
</table>
### Table of Standard Implicit (Suffix) Rules for Ratfor Files

<table>
<thead>
<tr>
<th>Implicit Rule Name</th>
<th>Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>.r~.a</td>
<td>$(GET) $(GFLAGS) -p $&lt; &gt; $*.r</td>
</tr>
<tr>
<td></td>
<td>$(COMPILE.r) -o % $*.r</td>
</tr>
<tr>
<td></td>
<td>$(AR) $(ARFLAGS) @ %</td>
</tr>
<tr>
<td></td>
<td>$(RM) %</td>
</tr>
</tbody>
</table>

### Table of Standard Implicit (Suffix) Rules for SCCS Files

<table>
<thead>
<tr>
<th>Implicit Rule Name</th>
<th>Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>.SCCS_GET</td>
<td>sccs $(SCCSFLAGS) get $(SCCSGETFLAGS) @ -G@</td>
</tr>
<tr>
<td>.SCCS_GET_POSIX</td>
<td>sccs $(SCCSFLAGS) get $(SCCSGETFLAGS) @</td>
</tr>
<tr>
<td>.GET_POSIX</td>
<td>$(GET) $(GFLAGS) s.@</td>
</tr>
</tbody>
</table>

### Table of Standard Implicit (Suffix) Rules for Shell Scripts

<table>
<thead>
<tr>
<th>Implicit Rule Name</th>
<th>Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>.sh</td>
<td>cat $&lt; &gt;@</td>
</tr>
<tr>
<td></td>
<td>chmod +x @</td>
</tr>
<tr>
<td>.sh~</td>
<td>$(GET) $(GFLAGS) -p $&lt; &gt;$*.sh</td>
</tr>
<tr>
<td></td>
<td>cp $*.sh @</td>
</tr>
<tr>
<td></td>
<td>chmod a+x @</td>
</tr>
</tbody>
</table>

### Table of Standard Implicit (Suffix) Rules for yacc Files

<table>
<thead>
<tr>
<th>Implicit Rule Name</th>
<th>Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>.y</td>
<td>$(YACC.y) $&lt;</td>
</tr>
<tr>
<td></td>
<td>$(LINK.c) -o @ y.tab.c $(LDLIBS)</td>
</tr>
<tr>
<td></td>
<td>$(RM) y.tab.c</td>
</tr>
</tbody>
</table>
## Table of Standard Implicit (Suffix) Rules for yacc Files

<table>
<thead>
<tr>
<th>Implicit Rule Name</th>
<th>Command Line</th>
</tr>
</thead>
</table>
| .y.c               | $(YACC.y) $<  
                   | mv y.tab.c $@  |
| .y.in              | $(YACC.y) $<  
                   | $(LINT.c) -o $@ -i y.tab.c  
                   | $(RM) y.tab.c  |
| .y.o               | $(YACC.y) $<  
                   | $(COMPILE.c) -o $@ y.tab.c  
                   | $(RM) y.tab.c  |
| .y~                | $(GET) $(GFLAGS) -p $< > $*.y  
                   | $(YACC) $(YFLAGS) $*.y  
                   | $(COMPILE.c) -o $@ y.tab.c  
                   | $(RM) y.tab.c  |
| .y~.c              | $(GET) $(GFLAGS) -p $< > $*.y  
                   | $(YACC) $(YFLAGS) $*.y  
                   | mv y.tab.c $@  |
| .y~.ln             | $(GET) $(GFLAGS) -p $< > $*.y  
                   | $(YACC.y) $*.y  
                   | $(LINT.c) -o $@ -i y.tab.c  
                   | $(RM) y.tab.c  |
| .y~.o              | $(GET) $(GFLAGS) -p $< > $*.y  
                   | $(YACC) $(YFLAGS) $*.y  |
Table of Standard Implicit (Suffix) Rules for yacc Files

<table>
<thead>
<tr>
<th>Implicit Rule Name</th>
<th>Command Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(CC) $(CFLAGS) -c y.tab.c</td>
<td></td>
</tr>
<tr>
<td>rm -f y.tab.c</td>
<td></td>
</tr>
<tr>
<td>mv y.tab.o $@</td>
<td></td>
</tr>
</tbody>
</table>

make reads in the standard set of implicit rules from the file
/usr/share/lib/make/make.rules, unless -r is in effect, or there is a
make.rules file in the local directory that does not include that file.

The Suffixes List

The suffixes list is given as the list of dependencies for the `'.
SUFFIXES:'
special-function target. The default list is contained in the SUFFIXES macro (See Table
of Predefined Macros for the standard list of suffixes). You can define additional
.SUFFIXES: targets; a .SUFFIXES target with no dependencies clears the list of
suffixes. Order is significant within the list; make selects a rule that corresponds to the
target’s suffix and the first dependency-file suffix found in the list. To place suffixes at
the head of the list, clear the list and replace it with the new suffixes, followed by the
default list:

.SUFFIXES:

A tilde (~) indicates that if a dependency file with the indicated suffix (minus the ~) is
under SCCS its most recent version should be retrieved, if necessary, before the target
is processed.

Library Maintenance

A target name of the form:

lib (member . . .)

refers to a member, or a space-separated list of members, in an ar(1) library.

The dependency of the library member on the corresponding file must be given as an
explicit entry in the makefile. This can be handled by a pattern matching rule of the
form:

lib (%s): %s

where .s is the suffix of the member; this suffix is typically .o for object libraries.

A target name of the form:

lib (symbol)

refers to the member of a randomized object library that defines the entry point named
symbol.

Command Execution

Command lines are executed one at a time, each by its own process or shell. Shell
commands, notably cd, are ineffectual across an unescaped NEWLINE in the makefile.
A line is printed (after macro expansion) just before being executed. This is suppressed
if it starts with a `@', if there is a `.SILENT:' entry in the makefile, or if make is run
with the -s option. Although the -n option specifies printing without execution, lines
containing the macro $(MAKE)$ are executed regardless, and lines containing the @ special character are printed. The -t (touch) option updates the modification date of a file without executing any rules. This can be dangerous when sources are maintained by more than one person.

`make` invokes the shell with the -e (exit-on-errors) argument. Thus, with semicolon-separated command sequences, execution of the later commands depends on the success of the former. This behavior can be overridden by starting the command line with a `-' or by writing a shell script that returns a non-zero status only as it finds appropriate.

To use the Bourne shell `if` control structure for branching, use a command line of the form:

```bash
if expression ; then command ; \ . . . ; else command ; \ . . . ; \fi
```

Although composed of several input lines, the escaped NEWLINE characters insure that `make` treats them all as one (shell) command line.

To use the Bourne shell `for` control structure for loops, use a command line of the form:

```bash
for var in list ; do command ; \ . . . ; done
```

To refer to a shell variable, use a double-dollar-sign (``$``). This prevents expansion of the dollar-sign by `make`.

To incorporate the standard output of a shell command in a macro, use a definition of the form:

```bash
MACRO :sh =command
```

The command is executed only once, standard error output is discarded, and NEWLINE characters are replaced with SPACEs. If the command has a non-zero exit status, `make` halts with an error.

To capture the output of a shell command in a macro reference, use a reference of the form:

```bash
$(MACRO :sh)
```

where `MACRO` is the name of a macro containing a valid Bourne shell command line. In this case, the command is executed whenever the reference is evaluated. As with shell command substitutions, the reference is replaced with the standard output of the command. If the command has a non-zero exit status, `make` halts with an error.

In contrast to commands in rules, the command is not subject for macro substitution; therefore, a dollar sign ($) need not be replaced with a double dollar sign ($$$).

**Signals**

`INT`, `SIGTERM`, and `QUIT` signals received from the keyboard halt `make` and remove the target file being processed unless that target is in the dependency list for `.PRECIOUS:`.
EXAMPLE 1 Defining dependencies

This makefile says that `pgm` depends on two files `a.o` and `b.o`, and that they in turn depend on their corresponding source files (`a.c` and `b.c`) along with a common file `incl.h`:

```
pgm: a.o b.o
   $(LINK.c) -o $@
a.o: incl.h a.c
    cc -c a.c
b.o: incl.h b.c
    cc -c b.c
```

EXAMPLE 2 Using implicit rules

The following makefile uses implicit rules to express the same dependencies:

```
pgm: a.o b.o
    cc a.o b.o -o pgma.o
b.o: incl.h
```

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of make: LC_CTYPE, LC_MESSAGES, and NLSPATH.

- **KEEP_STATE**
  
  This environment variable has the same effect as the `KEEP_STATE` special-function target. It enables command dependencies, hidden dependencies and writing of the state file.

- **USE_SVR4_MAKE**
  
  This environment variable causes make to invoke the generic System V version of make (`/usr/ccs/lib/svr4.make`). See sysV-make(1).

- **MAKEFLAGS**
  
  This variable is interpreted as a character string representing a series of option characters to be used as the default options. The implementation will accept both of the following formats (but need not accept them when intermixed):

  1. The characters are option letters without the leading hyphens or blank character separation used on a command line.

  2. The characters are formatted in a manner similar to a portion of the `make` command line: options are preceded by hyphens and blank-character-separated. The `macro=name` macro definition operands can also be included. The difference between the contents of `MAKEFLAGS` and the command line is that the contents of the variable will not be subjected to the word expansions (see wordexp(3C)) associated with parsing the command line values.
When the command-line options -f or -p are used, they will take effect regardless of whether they also appear in MAKEFLAGS. If they otherwise appear in MAKEFLAGS, the result is undefined.

The MAKEFLAGS variable will be accessed from the environment before the makefile is read. At that time, all of the options (except -f and -p) and command-line macros not already included in MAKEFLAGS are added to the MAKEFLAGS macro. The MAKEFLAGS macro will be passed into the environment as an environment variable for all child processes. If the MAKEFLAGS macro is subsequently set by the makefile, it replaces the MAKEFLAGS variable currently found in the environment.

EXIT STATUS

When the -q option is specified, the make utility will exit with one of the following values:

0 Successful completion.
1 The target was not up-to-date.
>1 An error occurred.

When the -q option is not specified, the make utility will exit with one of the following values:

0 Successful completion
>0 An error occurred

FILES

makefile
Makefile
    current version(s) of make description file
s.makefile
s.Makefile
    SCCS history files for the above makefile(s) in the current directory
SCCS/s.makefile
SCCS/s.Makefile
    SCCS history files for the above makefile(s)
make.rules
    default file for user-defined targets, macros, and implicit rules
/usr/share/lib/make/make.rules
    makefile for standard implicit rules and macros (not read if make.rules is).
.make.state
    state file in the local directory
make(1S)

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsprot</td>
</tr>
</tbody>
</table>

**SEE ALSO**

ar(1), arch(1), cd(1), cpp(1), lex(1), mach(1), sccs-get(1), sh(1), sysV-make(1), yacc(1), wordexp(3C), passwd(4), attributes(5), environ(5), POSIX.2(5)

*Solaris Advanced User's Guide*

**DIAGNOSTICS**

Don't know how to make target `target'

There is no makefile entry for `target', and none of `make`'s implicit rules apply (there is no dependency file with a suffix in the suffixes list, or the target's suffix is not in the list).

*** target removed.
make was interrupted while building `target'. Rather than leaving a partially-completed version that is newer than its dependencies, `make' removes the file named `target'.

*** target not removed.
make was interrupted while building `target' and `target' was not present in the directory.

*** target could not be removed, reason
make was interrupted while building `target', which was not removed for the indicated reason.

Read of include file `file' failed

The makefile indicated in an include directive was not found, or was inaccessible.

Loop detected when expanding macro value `macro'

A reference to the macro being defined was found in the definition.

Could not write state file `file'

You used the `.KEEP_STATE': target, but do not have write permission on the state file.

*** Error code n

The previous shell command returned a nonzero error code.

*** signal message

The previous shell command was aborted due to a signal. If `– core dumped' appears after the message, a core file was created.
Conditional macro conflict encountered

Displayed only when -d is in effect, this message indicates that two or more parallel targets currently being processed depend on a target which is built differently for each by virtue of conditional macros. Since the target cannot simultaneously satisfy both dependency relationships, it is conflicted.

Some commands return nonzero status inappropriately; to overcome this difficulty, prefix the offending command line in the rule with a ‘-’.

Filenames with the characters ‘=’, ‘:’, or ‘@’, do not work.

You cannot build file.o from lib(file.o).

Options supplied by MAKEFLAGS should be reported for nested make commands. Use the -d option to find out what options the nested command picks up from MAKEFLAGS.

This version of make is incompatible in certain respects with previous versions:

- The -d option output is much briefer in this version. -dd now produces the equivalent voluminous output.
- make attempts to derive values for the dynamic macros ‘$*’, ‘$<’, and ‘$?’; while processing explicit targets. It uses the same method as for implicit rules; in some cases this can lead either to unexpected values, or to an empty value being assigned. (Actually, this was true for earlier versions as well, even though the documentation stated otherwise.)
- make no longer searches for SCCS history "(s.)" files.
- Suffix replacement in macro references are now applied after the macro is expanded.

There is no guarantee that makefiles created for this version of make will work with earlier versions.

If there is no make.rules file in the current directory, and the file /usr/share/lib/make/make.rules is missing, make stops before processing any targets. To force make to run anyway, create an empty make.rules file in the current directory.

Once a dependency is made, make assumes the dependency file is present for the remainder of the run. If a rule subsequently removes that file and future targets depend on its existence, unexpected errors may result.

When hidden dependency checking is in effect, the $? macro’s value includes the names of hidden dependencies. This can lead to improper filename arguments to commands when $? is used in a rule.

Pattern replacement macro references cannot be used in the dependency list of a pattern matching rule.
Unlike previous versions, this version of `make` strips a leading `./` from the value of the `$@` dynamic macro.

With automatic SCCS retrieval, this version of `make` does not support tilde suffix rules.

The only dynamic macro whose value is strictly determined when used in a dependency list is `$@` (takes the form `$$@`).

`make` invokes the shell with the `-e` argument. This cannot be inferred from the syntax of the rule alone.
NAME

man – find and display reference manual pages

SYNOPSIS

man [ - ] [-adFlrt] [-M path] [-T macro-package] [-s section] name...

man [-M path] -k keyword...

man [-M path] -f file...

DESCRIPTION

The man command displays information from the reference manuals. It displays complete manual pages that you select by name, or one-line summaries selected either by keyword (-k), or by the name of an associated file (-f). If no manual page is located, man prints an error message.

Source Format

Reference Manual pages are marked up with either nroff(1) or sgml(5) (Standard Generalized Markup Language) tags. The man command recognizes the type of markup and processes the file accordingly. The various source files are kept in separate directories depending on the type of markup.

Location of Manual Pages

The online Reference Manual page directories are conventionally located in /usr/share/man. The nroff sources are located in the /usr/share/man/man* directories. The SGML sources are located in the /usr/share/man/sman* directories. Each directory corresponds to a section of the manual. Since these directories are optionally installed, they may not reside on your host. You may have to mount /usr/share/man from a host on which they do reside.

If there are preformatted, up-to-date versions in the corresponding cat* or fmt* directories, man simply displays or prints those versions. If the preformatted version of interest is out of date or missing, man reformats it prior to display and will store the preformatted version if cat* or fmt* is writable. The windex database is not updated. See catman(1M). If directories for the preformatted versions are not provided, man reformats a page whenever it is requested. man uses a temporary file to store the formatted text during display.

If the standard output is not a terminal, or if the ‘-‘ flag is given, man pipes its output through cat(1). Otherwise, man pipes its output through more(1) to handle paging and underlining on the screen.

OPTIONS

The following options are supported:

- a
  Shows all manual pages matching name within the MANPATH search path. Manual pages are displayed in the order found.

- d
  Debugs. Displays what a section-specifier evaluates to, method used for searching, and paths searched by man.

- f file...
  man attempts to locate manual pages related to any of the given files. It strips the leading path name components from each file, and then prints one-line summaries containing the resulting basename or names. This option also uses the windex database.

- F
  Forces man to search all directories specified by MANPATH or the man.cf file, rather than using the windex lookup database. This

| User Commands | 853 |

is useful if the database is not up to date. If the windex database does not exist, this option is assumed.

-\texttt{k keyword} . . . Prints out one-line summaries from the windex database (table of contents) that contain any of the given keywords. The windex database is created using \texttt{catman(1M)}.

-\texttt{l} Lists all manual pages found matching \texttt{name} within the search path.

-\texttt{M path} Specifies an alternate search path for manual pages. \texttt{path} is a colon-separated list of directories that contain manual page directory subtrees. For example, if \texttt{path} is \texttt{/usr/share/man:/usr/local/man}, \texttt{man} searches for \texttt{name} in the standard location, and then \texttt{/usr/local/man}. When used with the \texttt{-k} or \texttt{-f} options, the \texttt{-M} option must appear first. Each directory in the \texttt{path} is assumed to contain subdirectories of the form \texttt{man*} or \texttt{sman*}, one for each section. This option overrides the \texttt{MANPATH} environment variable.

-\texttt{r} Reformats the manual page, but does not display it. This replaces the \texttt{man - -t name} combination.

-\texttt{s section} . . . Specifies sections of the manual for \texttt{man} to search. The directories searched for \texttt{name} are limited to those specified by \texttt{section}. \texttt{section} can be a numerical digit (perhaps followed by one or more letters to match the desired section of the manual, for example, “3libucb”); a word (for example: local, new, old, public); or a letter. To specify multiple sections, separate each section with a comma. This option overrides the \texttt{MANPATH} environment variable and the \texttt{man.cf} file. See Search Path below for an explanation of how \texttt{man} conducts its search.

-\texttt{t} \texttt{man} arranges for the specified manual pages to be troffed to a suitable raster output device (see \texttt{troff(1)}). If both the \texttt{-} and \texttt{-t} flags are given, \texttt{man} updates the troffed versions of each named \texttt{name} (if necessary), but does not display them.

-\texttt{T macro-package} Formats manual pages using \texttt{macro-package} rather than the standard -\texttt{man} macros defined in \texttt{/usr/share/lib/tmac/an}. See Search Path under USAGE for a complete explanation of the default search path order.

\textbf{OPERANDS} The following operand is supported:

\texttt{name} A keyword or the name of a standard utility.

\textbf{Manual Page Sections} Entries in the reference manuals are organized into sections. A section name consists of a major section name, typically a single digit, optionally followed by a subsection name, typically one or more letters. An unadorned major section name, for example, “9”, does not act as an abbreviation for the subsections of that name, such as “9e”, 854 man pages section 1: User Commands • Last Revised 10 Dec 2001
"9f", or "9s". That is, each subsection must be searched separately by `man -s`. Each section contains descriptions apropos to a particular reference category, with subsections refining these distinctions. See the `intro` manual pages for an explanation of the classification used in this release.

**Search Path**

Before searching for a given `name`, `man` constructs a list of candidate directories and sections. `man` searches for `name` in the directories specified by the `MANPATH` environment variable. If this variable is not set, `/usr/share/man` is searched by default.

Within the manual page directories, `man` confines its search to the sections specified in the following order:

- sections specified on the command line with the `-s` option
- sections embedded in the `MANPATH` environment variable
- sections specified in the `man.cf` file for each directory specified in the `MANPATH` environment variable

If none of the above exist, `man` searches each directory in the manual page path, and displays the first matching manual page found.

The `man.cf` file has the following format:

```
MANSECTIONS=section[.section]...
```

Lines beginning with `#` and blank lines are considered comments, and are ignored. Each directory specified in `MANPATH` can contain a manual page configuration file, specifying the default search order for that directory.

**Formatting Manual Pages**

Manual pages are marked up in `nroff(1)` or `sgml(5)`. `Nroff` manual pages are processed by `nroff(1)` or `troff(1)` with the `-man` macro package. Please refer to `man(5)` for information on macro usage. `SGML`—tagged manual pages are processed by an `SGML` parser and passed to the formatter.

When formatting an `nroff` manual page, `man` examines the first line to determine whether it requires special processing. If the first line is a string of the form:

```
\"X
```

where `X` is separated from the `"` by a single SPACE and consists of any combination of characters in the following list, `man` pipes its input to `troff(1)` or `nroff(1)` through the corresponding preprocessors.

- `e` `eqn(1)`, or `neqn` for `nroff`
- `r` `refer(1)`
- `t` `tbl(1)`
- `v` `vgrind(1)`
If eqn or neqn is invoked, it will automatically read the file /usr/pub/eqnchar (see eqnchar(5)). If nroff(1) is invoked, col(1) is automatically used.

If the first line of the nroff manual page is a reference to another manual page entry fitting the pattern:

```
.so man*/sourcefile
```

man processes the indicated file in place of the current one. The reference must be expressed as a path name relative to the root of the manual page directory subtree.

When the second or any subsequent line starts with .so, man ignores it; troff(1) or nroff(1) processes the request in the usual manner.

Manual pages are identified as being marked up in SGML by the presence of the string <!DOCTYPE . If the file also contains the string SHADOW_PAGE, the file refers to another manual page for the content. The reference is made with a file entity reference to the manual page that contains the text. This is similar to the .so mechanism used in the nroff formatted man pages.

See environ(5) for descriptions of the following environment variables that affect the execution of man: LC_CTYPE, LC_MESSAGES, and NLSPATH.

**MANPATH**
A colon-separated list of directories; each directory can be followed by a comma-separated list of sections. If set, its value overrides /usr/share/man as the default directory search path, and the man.cf file as the default section search path. The -M and -s flags, in turn, override these values.

**PAGER**
A program to use for interactively delivering man’s output to the screen. If not set, ‘more -s’ is used. See more(1).

**TCAT**
The name of the program to use to display troffed manual pages.

**TROFF**
The name of the formatter to use when the -t flag is given. If not set, troff(1) is used.

The following exit values are returned:

- **0**  
  Successful completion.

- **>0**  
  An error occurred.

**FILES**

- /usr/share/man
  root of the standard manual page directory subtree

- /usr/share/man/man?/*
  unformatted nroff manual entries

- /usr/share/man/sman?/*
  unformatted SGML manual entries
ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWdoc</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled (see NOTES)</td>
</tr>
</tbody>
</table>

SEE ALSO

apropos(1), cat(1), col(1), eqn(1), more(1), nroff(1), refer(1), tbl(1), troff(1), vgrind(1), whatis(1), catman(1M), attributes(5), environ(5), eqnchar(5), man(5), sgml(5)

NOTES

The -f and -k options use the windex database, which is created by catman(1M).

The man command is CSI-capable. However, some utilities invoked by the man command, namely, troff, eqn, neqn, refer, tbl, and vgrind, are not verified to be CSI-capable. Because of this, the man command with the -t option may not handle non-EUC data. Also, using the man command to display man pages that require special processing through eqn, neqn, refer, tbl, or vgrind may not be CSI-capable.

BUGS

The manual is supposed to be reproducible either on a phototypesetter or on an ASCII terminal. However, on a terminal some information (indicated by font changes, for instance) is lost.
Some dumb terminals cannot process the vertical motions produced by the e (see eqn(1)) preprocessing flag. To prevent garbled output on these terminals, when you use e, also use t, to invoke col(1) implicitly. This workaround has the disadvantage of eliminating superscripts and subscripts, even on those terminals that can display them. Control-q will clear a terminal that gets confused by eqn(1) output.
NAME
mconnect – connect to SMTP mail server socket

SYNOPSIS
mconnect [-p port] [-r] [hostname]

DESCRIPTION
The mconnect utility opens a connection to the mail server on a given host, so that it can be tested independently of all other mail software. If no host is given, the connection is made to the local host. Servers expect to speak the Simple Mail Transfer Protocol (SMTP) on this connection. Exit by typing the quit command. Typing EOF sends an end of file to the server. An interrupt closes the connection immediately and exits.

OPTIONS
The following options are supported:

- p port Specify the port number instead of the default SMTP port (number 25) as the next argument.
- r "Raw" mode: disable the default line buffering and input handling. This produces an effect similar to telnet(1) to port number 25.

OPERANDS
The following operand is supported:

hostname The name of a given host.

USAGE
The mconnect command is IPv6-enabled. See ip6(7P).

FILES
/etc/mail/sendmail.hf help file for SMTP commands

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO
telnet(1), sendmail(1M), attributes(5), ip6(7P)

Postel, Jonathan B., RFC 821, Simple Mail Transfer Protocol, Information Sciences Institute, University of Southern California, August 1982.
mcs(1)

NAME  mcs – manipulate the comment section of an object file

SYNOPSIS  /usr/ccs/bin/mcs { -c | -d | -p | -V | -a string | -n name...}file...

DESCRIPTION  The mcs command is used to manipulate a section, by default the .comment section, in an ELF object file. It is used to add to, delete, print, and compress the contents of a section in an ELF object file, and print only the contents of a section in a COFF object file. mcs cannot add, delete, or compress the contents of a section that is contained within a segment.

If the input file is an archive (see ar(3HEAD)), the archive is treated as a set of individual files. For example, if the -a option is specified, the string is appended to the comment section of each ELF object file in the archive; if the archive member is not an ELF object file, then it is left unchanged.

mcs must be given one or more of the options described below. It applies, in order, each of the specified options to each file.

OPTIONS  The following options are supported:

- a string  Appends string to the comment section of the ELF object files. If string contains embedded blanks, it must be enclosed in quotation marks.

- c  Compresses the contents of the comment section of the ELF object files. All duplicate entries are removed. The ordering of the remaining entries is not disturbed.

- d  Deletes the contents of the comment section from the ELF object files. The section header for the comment section is also removed.

- n name  Specifies the name of the comment section to access if other than .comment. By default, mcs deals with the section named .comment. This option can be used to specify another section. mcs can take multiple -n options to allow for specification of multiple section comments.

- p  Prints the contents of the comment section on the standard output. Each section printed is tagged by the name of the file from which it was extracted, using the format file [member_name] : for archive files and file : for other files.

- V  Prints on standard error the version number of mcs.

EXAMPLES  EXAMPLE 1 Printing a file’s comment section

The following entry

example$ /usr/ccs/bin/mcs -p elf.file prints the comment section of the file elf.file.
EXAMPLE 2 Appending a string to a comment section

The following entry

eexample> /usr/ccs/bin/mcs -a xyz elf.file

appends string xyz to elf.file’s comment section.

FILES
/tmp/mcs* temporary files

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWbtool</td>
</tr>
</tbody>
</table>

SEE ALSO
ar(1), as(1), ld(1), elf(3ELF), tmpnam(3C), a.out(4), ar(3HEAD), attributes(5)

NOTES
When mcs deletes a section using the -d option, it tries to bind together sections of type SHT_REL and target sections pointed to by the sh_info section header field. If one is to be deleted, mcs attempts to delete the other of the pair.
mdb(1)

NAME  mdb – modular debugger


Introduction  The mdb utility is an extensible utility for low-level debugging and editing of the live operating system, operating system crash dumps, user processes, user process core dumps, and object files. For a more detailed description of mdb features, refer to the manual, Solaris Modular Debugger Guide.

Debugging is the process of analyzing the execution and state of a software program in order to remove defects. Traditional debugging tools provide facilities for execution control so that programmers can re-execute programs in a controlled environment and display the current state of program data or evaluate expressions in the source language used to develop the program.

Unfortunately, these techniques are often inappropriate for debugging complex software systems such as an operating system, where bugs may not be reproducible and program state is massive and distributed, for programs that are highly optimized, have had their debug information removed, or are themselves low-level debugging tools, or for customer situations where the developer can only access post-mortem information.

mdb provides a completely customizable environment for debugging these programs and scenarios, including a dynamic module facility that programmers can use to implement their own debugging commands to perform program-specific analysis. Each mdb module can be used to examine the program in several different contexts, including live and post-mortem.

Definitions  The target is the program being inspected by the debugger. mdb currently provides support for the following types of targets: user processes, user process core files, the live operating system (via /dev/kmem and /dev/ksyms), operating system crash dumps, user process images recorded inside an operating system crash dump, ELF object files, and raw binary files. Each target exports a standard set of properties, including one or more address spaces, one or more symbol tables, a set of load objects, and a set of threads that can be examined using the debugger commands described below.

A debugger command, or dcmd (pronounced dee-command) in mdb terminology, is a routine in the debugger that can access any of the properties of the current target. mdb parses commands from standard input, and then executes the corresponding dcmds. Each dcmd can also accept a list of string or numerical arguments, as shown in the syntax description below. mdb contains a set of built-in dcmds, described below, that are always available. You can also extend the capabilities of mdb itself by writing your own dcmds, as described in the Solaris Modular Debugger Guide.
A walker is a set of routines that describe how to walk, or iterate, through the elements of a particular program data structure. A walker encapsulates the data structure’s implementation from dcmds and from mdb itself. You can use walkers interactively, or use them as a primitive to build other dcmds or walkers. As with dcmds, you can extend mdb by implementing your own walkers as part of a debugger module.

A debugger module, or dmod (pronounced dee-mod), is a dynamically loaded library containing a set of dcmds and walkers. During initialization, mdb will attempt to load dmods corresponding to the load objects present in the target. You can subsequently load or unload dmods at any time while running mdb. mdb ships with a set of standard dmods for debugging the Solaris kernel. The Solaris Modular Debugger Guide contains more information on developing your own debugger modules.

A macro file is a text file containing a set of commands to execute. Macro files are typically used to automate the process of displaying a simple data structure. mdb provides complete backward compatibility for the execution of macro files written for adb(1), and the Solaris installation includes a set of macro files for debugging the Solaris kernel that may be used with either tool.

The debugger processes commands from standard input. If standard input is a terminal, mdb provides terminal editing capabilities. mdb can also process commands from macro files and from dcmd pipelines, described below. The language syntax is designed around the concept of computing the value of an expression (typically a memory address in the target), and then applying a dcmd to that address. The current address location is referred to as dot, and its value is referenced using "".

A metacharacter is one of the following characters:

```
[ ] | ! / \ ? = > $ : ;
NEWLINE SPACE TAB
```

A blank is a TAB or a SPACE. A word is a sequence of characters separated by one or more non-quoted metacharacters. Some of the metacharacters only function as delimiters in certain contexts, as described below. An identifier is a sequence of letters, digits, underscores, periods, or backquotes beginning with a letter, underscore, or period. Identifiers are used as the names of symbols, variables, dcmds, and walkers. Commands are delimited by a NEWLINE or semicolon ( ; ).

A dcmd is denoted by one of the following words or metacharacters:

```
/ \ ? = > $character :character ::identifier
```

dcmds named by metacharacters or prefixed by a single $ or : are provided as built-in operators, and implement complete compatibility with the command set of the legacy adb(1) utility. Once a dcmd has been parsed, the /, \, ?, =, >, $, and : characters are no longer recognized as metacharacters until the termination of the argument list.

User Commands  863
A simple-command is a dcmd followed by a sequence of zero or more blank-separated words. The words are passed as arguments to the invoked dcmd, except as specified under Quoting and Arithmetic Expansion below. Each dcmd returns an exit status that indicates it was either successful, failed, or was invoked with invalid arguments.

A pipeline is a sequence of one or more simple commands separated by |. Unlike the shell, dcmds in mdb pipelines are not executed as separate processes. After the pipeline has been parsed, each dcmd is invoked in order from left to right. Each dcmd’s output is processed and stored as described under dcmd Pipelines below. Once the left-hand dcmd is complete, its processed output is used as input for the next dcmd in the pipeline. If any dcmd does not return a successful exit status, the pipeline is aborted.

An expression is a sequence of words that is evaluated to compute a 64-bit unsigned integer value. The words are evaluated using the rules described under Arithmetic Expansion below.

### Commands

A command is one of the following:

- **pipeline** [ ! word ... ] [ ; ]
  
  A simple-command or pipeline can be optionally suffixed with the ! character, indicating that the debugger should open a pipe(2) and send the standard output of the last dcmd in the mdb pipeline to an external process created by executing $SHELL -c followed by the string formed by concatenating the words after the ! character. For more details, refer to Shell Escapes below.

- **expression pipeline** [ ! word ... ] [ ; ]
  
  A simple-command or pipeline can be prefixed with an expression. Before execution of the pipeline, the value of dot (the variable denoted by ".") is set to the value of the expression.

- **expression** , **expression pipeline** [ ! word ... ] [ ; ]
  
  A simple-command or pipeline can be prefixed with two expressions. The first is evaluated to determine the new value of dot, and the second is evaluated to determine a repeat count for the first dcmd in the pipeline. This dcmd will be executed count times before the next dcmd in the pipeline is executed. The repeat count only applies to the first dcmd in the pipeline.

- **expression pipeline** [ ! word ... ] [ ; ]
  
  If the initial expression is omitted, dot is not modified but the first dcmd in the pipeline will be repeated according to the value of the expression.

- **expression** [ ! word ... ] [ ; ]
  
  A command can consist only of an arithmetic expression. The expression is evaluated and the dot variable is set to its value, and then the previous dcmd and arguments are executed using the new value of dot.
A command can consist only of a dot expression and repeat count expression. After dot is set to the value of the first expression, the previous dcmd and arguments are repeatedly executed the number of times specified by the value of the second expression.

If the initial expression is omitted, dot is not modified but the previous dcmd and arguments are repeatedly executed the number of times specified by the value of the count expression.

If the command begins with the ! character, no dcmds are executed and the debugger simply executes $SHELL -c followed by the string formed by concatenating the words after the ! character.

A word beginning with // causes that word and all the subsequent characters up to a NEWLINE to be ignored.

Arithmetic expansion is performed when an mdb command is preceded by an optional expression representing a start address, or a start address and a repeat count. Arithmetic expansion can also be performed to compute a numerical argument for a dcmd. An arithmetic expression can appear in an argument list enclosed in square brackets preceded by a dollar sign ($[expression]) and will be replaced by the value of the expression.

Expressions may contain any of the following special words:

- **integer**: The specified integer value. Integer values may be prefixed with 0i or 0I to indicate binary values, 0o or 0O to indicate octal values, 0t or 0T to indicate decimal values, and 0x or 0X to indicate hexadecimal values (the default).

- **0[1T][0-9]+.[0-9]+**: The specified decimal floating point value, converted to its IEEE double-precision floating point representation.

- **'cccccccc'**: The integer value computed by converting each character to a byte equal to its ASCII value. Up to eight characters may be specified in a character constant. Characters are packed into the integer in reverse order (right-to-left) beginning at the least significant byte.

- **<identifier****: The value of the variable named by identifier.

- **identifier**: The value of the symbol named by identifier.

- **(expression)**: The value of expression.

- **.**: The value of dot.

- **&**: The most recent value of dot used to execute a dcmd.
The value of dot incremented by the current increment.

The value of dot decremented by the current increment.

The increment is a global variable that stores the total bytes read by the last formatting dcmd. For more information on the increment, refer to the discussion of Formatting dcmds below.

Unary operators are right associative and have higher precedence than binary operators. The unary operators are:

#expression Logical negation.
~expression Bitwise complement.
-expression Integer negation.
%expression The value of a pointer-sized quantity at the object file location corresponding to virtual address expression in the target’s virtual address space.
%/[csil]/expression The value of a char, short, int, or long-sized quantity at the object file location corresponding to virtual address expression in the target’s virtual address space.
%/[1248]/expression The value of a one, two, four, or eight-byte quantity at the object file location corresponding to virtual address expression in the target’s virtual address space.
*expression The value of a pointer-sized quantity at virtual address expression in the target’s virtual address space.
*/[csil]/expression The value of a char, short, int, or long-sized quantity at virtual address expression in the target’s virtual address space.
*/[1248]/expression The value of a one, two, four, or eight-byte quantity at virtual address expression in the target’s virtual address space.

Binary operators are left associative and have lower precedence than unary operators. The binary operators, in order of precedence from highest to lowest, are:

* Integer multiplication.
% Integer division.
# Left-hand side rounded up to next multiple of right-hand side.
+ Integer addition.
- Integer subtraction.
<< Bitwise shift left.
Bitwise shift right.

== Logical equality.

!= Logical inequality.

& Bitwise AND.

^ Bitwise exclusive OR.

| Bitwise inclusive OR.

Quoting

Each metacharacter described above (see Syntax) terminates a word unless quoted. Characters can be quoted (forcing mdb to interpret each character as itself without any special significance) by enclosing them in a pair of single ('') or double (""") quote marks. A single quote cannot appear within single quotes. Inside double quotes, mdb recognizes the C programming language character escape sequences.

Shell Escapes

The ` character can be used to create a pipeline between an mdb command and the user's shell. If the SHELL environment variable is set, mdb will fork and exec this program for shell escapes; otherwise /bin/sh is used. The shell is invoked with the -c option followed by a string formed by concatenating the words after the ` character. The ` character takes precedence over all other metacharacters, except semicolon (;) and NEWLINE. Once a shell escape is detected, the remaining characters up to the next semicolon or NEWLINE are passed as is to the shell. The output of shell commands may not be piped to mdb dcmds. Commands executed by a shell escape have their output sent directly to the terminal, not to mdb.

Variables

A variable is a variable name, a corresponding integer value, and a set of attributes. A variable name is a sequence of letters, digits, underscores, or periods. A variable can be assigned a value using the > dcmd or ::typeset dcmd, and its attributes can be manipulated using the ::typeset dcmd. Each variable’s value is represented as a 64-bit unsigned integer. A variable may have one or more of the following attributes: read-only (cannot be modified by the user), persistent (cannot be unset by the user), and tagged (user-defined indicator).

The following variables are defined as persistent:

0 The most recent value printed using the /, ?, or = dcmd.

9 The most recent count used with the $< dcmd.

b The virtual address of the base of the data section.

d The size of the data section in bytes.

e The virtual address of the entry point.

m The initial bytes (magic number) of the target's primary object file, or zero if no object file has been read yet.

The size of the text section in bytes.
hits The count of the number of times the matched software event specifier has been matched. See Event Callbacks, below.

thread The thread identifier of the current representative thread. The value of the identifier depends on the threading model used by the current target. See Thread Support, below.

In addition, the mdb kernel and process targets will export the current values of the representative thread’s register set as named variables. The names of these variables will depend on the target’s platform and instruction set architecture.

As explained in the Syntax description above, a symbol identifier present in an expression context evaluates to the value of this symbol. The value typically denotes the virtual address of the storage associated with the symbol in the target’s virtual address space. A target may support multiple symbol tables including, but not limited to, a primary executable symbol table, a primary dynamic symbol table, a run-time link-editor symbol table, and standard and dynamic symbol tables for each of a number of load objects (such as shared libraries in a user process, or kernel modules in the Solaris kernel). The target typically searches the primary executable’s symbol tables first, and then one or more of the other symbol tables. Notice that ELF symbol tables only contain entries for external, global, and static symbols; automatic symbols do not appear in the symbol tables processed by mdb.

Additionally, mdb provides a private user-defined symbol table that is searched prior to any of the target symbol tables. The private symbol table is initially empty, and can be manipulated using the ::nmadd and ::nmdel dcmds. The ::nm -P option can be used to display the contents of the private symbol table. The private symbol table allows the user to create symbol definitions for program functions or data that were either missing from the original program or stripped out. These definitions are then used whenever mdb converts a symbolic name to an address, or an address to the nearest symbol.

As targets contain multiple symbol tables, and each symbol table may include symbols from multiple object files, different symbols with the same name may exist. mdb uses the backquote (`) character as a symbol name scoping operator to allow the programmer to obtain the value of the desired symbol in this situation. The programmer can specify the scope used to resolve a symbol name as either: object `name, or file `name, or object `file`name. The object identifier refers to the name of a load object. The file identifier refers to the basename of a source file that has a symbol of type STT_FILE in the specified object’s symbol table. The object identifier’s interpretation depends on the target type.

The mdb kernel target expects object to specify the basename of a loaded kernel module. For example, the symbol name

```
specfs`_init
```

evaluates to the value of the _init symbol in the specfs kernel module.
The `mdb` process target expects object to specify the name of the executable or of a loaded shared library. It may take any of the following forms:

1. An exact match (that is, a full pathname): `/usr/lib/libc.so.1`
2. An exact basename match: `libc.so.1`
3. An initial basename match up to a "." suffix: `libc.so` or `libc`
4. The literal string `a.out` is accepted as an alias for the executable.

The process target will also accept any of the four forms described above preceded by an optional link-map id (lmid). The lmid prefix is specified by an initial "LM" followed by the link-map id in hexadecimal followed by an additional backquote. For example, the symbol name

```
LM0'libc.so.1'_init
```

will evaluate to the value of the _init symbol in the `libc.so.1` library that is loaded on link-map 0 (LM_ID_BASE). The link-map specifier may be necessary to resolve symbol naming conflicts in the event that the same library is loaded on more than one link map. For more information on link maps, refer to the Linker and Libraries Guide and `dlopen(3DL)`. Link-map identifiers will be displayed when symbols are printed according to the setting of the `showlmid` option, as described under OPTIONS, below.

In the case of a naming conflict between symbols and hexadecimal integer values, `mdb` will attempt to evaluate an ambiguous token as a symbol first, before evaluating it as an integer value. For example, the token `f` may either refer to the decimal integer value 15 specified in hexadecimal (the default base), or to a global variable named `f` in the target's symbol table. If a symbol with an ambiguous name is present, the integer value can be specified by using an explicit `0x` or `0X` prefix.

As described earlier, each `mdb` dmod provides a set of dcmds and walkers. dcmds and walkers are tracked in two distinct, global namespaces. `mdb` also keeps track of a dcmd and walker namespace associated with each dmod. Identically named dcmds or walkers within a given dmod are not allowed: a dmod with this type of naming conflict will fail to load. Name conflicts between dcmds or walkers from different dmods are allowed in the global namespace. In the case of a conflict, the first dcmd or walker with that particular name to be loaded is given precedence in the global namespace. Alternate definitions are kept in a list in load order. The backquote character (`'`) may be used in a dcmd or walker name as a scoping operator to select an alternate definition. For example, if dmods m1 and m2 each provide a dcmd `d`, and m1 is loaded prior to m2, then:

```
::d Executes m1's definition of d.
::m1'd Executes m1's definition of d.
::m2'd Executes m2's definition of d.
```
If module \texttt{m1} were now unloaded, the next \texttt{dcmd} on the global definition list (\texttt{m2} \texttt{'d}) would be promoted to global visibility. The current definition of a \texttt{dcmd} or walker can be determined using the ::\texttt{which} \texttt{dcmd}, described below. The global definition list can be displayed using the ::\texttt{which -v} option.

\textbf{\texttt{dcmd} Pipelines}

\texttt{dcmds} can be composed into a pipeline using the | operator. The purpose of a pipeline is to pass a list of values, typically virtual addresses, from one \texttt{dcmd} or walker to another. Pipeline stages might be used to map a pointer from one type of data structure to a pointer to a corresponding data structure, to sort a list of addresses, or to select the addresses of structures with certain properties.

\texttt{mdb} executes each \texttt{dcmd} in the pipeline in order from left to right. The leftmost \texttt{dcmd} is executed using the current value of dot, or using the value specified by an explicit expression at the start of the command. When a | operator is encountered, \texttt{mdb} creates a pipe (a shared buffer) between the output of the \texttt{dcmd} to its left and the \texttt{mdb} parser, and an empty list of values. As the \texttt{dcmd} executes, its standard output is placed in the pipe and then consumed and evaluated by the parser, as if \texttt{mdb} were reading this data from standard input. Each line must consist of an arithmetic expression terminated by a \texttt{NEWLINE} or semicolon (;). The value of the expression is appended to the list of values associated with the pipe. If a syntax error is detected, the pipeline is aborted.

When the \texttt{dcmd} to the left of a | operator completes, the list of values associated with the pipe is then used to invoke the \texttt{dcmd} to the right of the | operator. For each value in the list, dot is set to this value and the right-hand \texttt{dcmd} is executed. Only the rightmost \texttt{dcmd} in the pipeline has its output printed to standard output. If any \texttt{dcmd} in the pipeline produces output to standard error, these messages are printed directly to standard error and are not processed as part of the pipeline.

\textbf{Signal Handling}

The debugger ignores the PIPE and QUIT signals. The INT signal aborts the command that is currently executing. The debugger intercepts and provides special handling for the ILL, TRAP, EMT, FPE, BUS, and SEGV signals. If any of these signals are generated asynchronously (that is, delivered from another process using \texttt{kill(2)}), \texttt{mdb} will restore the signal to its default disposition and dump core. However, if any of these signals are generated synchronously by the debugger process itself and a \texttt{dcmd} from an externally loaded \texttt{dmod} is currently executing, and standard input is a terminal, \texttt{mdb} will provide a menu of choices allowing the user to force a core dump, quit without producing a core dump, stop for attach by a debugger, or attempt to resume. The resume option will abort all active commands and unload the \texttt{dmod} whose \texttt{dcmd} was active at the time the fault occurred. It can then be subsequently re-loaded by the user. The resume option provides limited protection against buggy \texttt{dcmds}. Refer to WARNINGS, Use of the Error Recovery Mechanism, below for information about the risks associated with the resume option.

\textbf{Command Re-entry}

The text of the last \texttt{HISTSIZE} (default 128) commands entered from a terminal device are saved in memory. The in-line editing facility, described next, provides key mappings for searching and fetching elements from the history list.
If standard input is a terminal device, `mdb` provides some simple emacs-style facilities for editing the command line. The `search`, `previous`, and `next` commands in edit mode provide access to the history list. Only strings, not patterns, are matched when searching. In the table below, the notation for control characters is caret (^) followed by a character shown in upper case. The notation for escape sequences is `M-` followed by a character. For example, `M-f` (pronounced meta-eff) is entered by depressing ESC followed by 'f', or by depressing Meta followed by 'f' on keyboards that support a Meta key. A command line is committed and executed using `RETURN` or `NEWLINE`.

The edit commands are:

- `^F` Move cursor forward (right) one character.
- `M-f` Move cursor forward one word.
- `^B` Move cursor backward (left) one character.
- `M-b` Move cursor backward one word.
- `^A` Move cursor to start of line.
- `^E` Move cursor to end of line.
- `^D` Delete current character, if the current line is not empty. If the current line is empty, `^D` denotes EOF and the debugger will exit.
- `M-^H` (Meta-backspace) Delete previous word.
- `^K` Delete from the cursor to the end of the line.
- `^L` Clear the screen and reprint the current line.
- `^T` Transpose current character with next character.
- `^N` Fetch the next command from the history. Each time `^N` is entered, the next command forward in time is retrieved.
- `^P` Fetch the previous command from the history. Each time `^P` is entered, the next command backward in time is retrieved.
- `^R[string]` Search backward in the history for a previous command line containing `string`. The string should be terminated by a `RETURN` or `NEWLINE`. If `string` is omitted, the previous history element containing the most recent string is retrieved.

The editing mode also interprets the following user-defined sequences as editing commands. User defined sequences can be read or modified using the `stty` command.

- `erase` User defined erase character (usually `^H` or `^?`). Delete previous character.
- `intr` User defined interrupt character (usually `^C`). Abort the current command and print a new prompt.
kill  User defined kill character (usually ^U). Kill the entire current
    command line.
quit  User defined quit character (usually ^\). Quit the debugger.
suspend User defined suspend character (usually ^Z). Suspend the
debugger.
werase User defined word erase character (usually ^W). Erase the
    preceding word.

On keyboards that support an extended keypad with arrow keys, mdb will interpret
these keystrokes as editing commands:
up-arrow Fetch the previous command from the history (same as ^P).
down-arrow Fetch the next command from the history (same as ^N).
left-arrow Move cursor backward one character (same as ^B).
right-arrow Move cursor forward one character (same as ^F).

Output Pager
mdb provides a built-in output pager. The output pager is enabled if the debugger’s
standard output is a terminal device. Each time a command is executed, mdb will
pause after one screenful of output is produced and will display a pager prompt:

    >> More [<space>, <cr>, q, n, c, a] ?

The following key sequences are recognized by the pager:
SPACE Display the next screenful of output.
a, A Abort the current top-level command and return to the
    prompt.
c, C Continue displaying output without pausing at each
    screenful until the current top-level command is
    complete.
n, N, NEWLINE, RETURN Display the next line of output.
q, Q, ^C, ^\ Quit (abort) the current dcmd only.

Formatting dcmds
The /, \, ?, and = metacharacters are used to denote the special output formatting
dcmds. Each of these dcmds accepts an argument list consisting of one or more format
characters, repeat counts, or quoted strings. A format character is one of the ASCII
caracters shown in the table below. Format characters are used to read and format
data from the target. A repeat count is a positive integer preceding the format
character that is always interpreted in base 10 (decimal). A repeat count may also be
specified as an expression enclosed in square brackets preceded by a dollar sign ($ [ ]).
A string argument must be enclosed in double-quotes (" "). No blanks are
necessary between format arguments.

The formatting dcmds are:
Display data from the target’s virtual address space starting at the virtual address specified by dot.

\ Display data from the target’s physical address space starting at the physical address specified by dot.

? Display data from the target’s primary object file starting at the object file location corresponding to the virtual address specified by dot.

= Display the value of dot itself in each of the specified data formats. The = dcmd is therefore useful for converting between bases and performing arithmetic.

In addition to dot, mdb keeps track of another global value called the increment. The increment represents the distance between dot and the address following all the data read by the last formatting dcmd. For example, if a formatting dcmd is executed with dot equal to address A, and displays a 4-byte integer, then after this dcmd completes, dot is still A, but the increment is set to 4. The + character (described under Arithmetic Expansion above) would now evaluate to the value A + 4, and could be used to reset dot to the address of the next data object for a subsequent dcmd.

Most format characters increase the value of the increment by the number of bytes corresponding to the size of the data format, shown in the table. The table of format characters can be displayed from within mdb using the ::formats dcmd. The format characters are:

+ increment dot by the count (variable size)
- decrement dot by the count (variable size)
B hexadecimal int (1 byte)
C character using C character notation (1 byte)
D decimal signed int (4 bytes)
E decimal unsigned long long (8 bytes)
F double (8 bytes)
G octal unsigned long long (8 bytes)
H swap bytes and shorts (4 bytes)
I address and disassembled instruction (variable size)
J hexadecimal long long (8 bytes)
K hexadecimal uintptr_t (4 or 8 bytes)
N newline
O octal unsigned int (4 bytes)
## mdb(1)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>symbol (4 or 8 bytes)</td>
</tr>
<tr>
<td>Q</td>
<td>octal signed int (4 bytes)</td>
</tr>
<tr>
<td>R</td>
<td>binary int (8 bytes)</td>
</tr>
<tr>
<td>S</td>
<td>string using C string notation (variable size)</td>
</tr>
<tr>
<td>T</td>
<td>horizontal tab</td>
</tr>
<tr>
<td>U</td>
<td>decimal unsigned int (4 bytes)</td>
</tr>
<tr>
<td>V</td>
<td>decimal unsigned int (1 byte)</td>
</tr>
<tr>
<td>W</td>
<td>default radix unsigned int (4 bytes)</td>
</tr>
<tr>
<td>X</td>
<td>hexadecimal int (4 bytes)</td>
</tr>
<tr>
<td>Y</td>
<td>decoded time32_t (4 bytes)</td>
</tr>
<tr>
<td>Z</td>
<td>hexadecimal long long (8 bytes)</td>
</tr>
<tr>
<td>^</td>
<td>decrement dot by increment * count (variable size)</td>
</tr>
<tr>
<td>a</td>
<td>dot as symbol+offset</td>
</tr>
<tr>
<td>b</td>
<td>octal unsigned int (1 byte)</td>
</tr>
<tr>
<td>c</td>
<td>character (1 byte)</td>
</tr>
<tr>
<td>d</td>
<td>decimal signed short (2 bytes)</td>
</tr>
<tr>
<td>e</td>
<td>decimal signed long long (8 bytes)</td>
</tr>
<tr>
<td>f</td>
<td>float (4 bytes)</td>
</tr>
<tr>
<td>g</td>
<td>octal signed long long (8 bytes)</td>
</tr>
<tr>
<td>h</td>
<td>swap bytes (2 bytes)</td>
</tr>
<tr>
<td>i</td>
<td>disassembled instruction (variable size)</td>
</tr>
<tr>
<td>n</td>
<td>newline</td>
</tr>
<tr>
<td>o</td>
<td>octal unsigned short (2 bytes)</td>
</tr>
<tr>
<td>p</td>
<td>symbol (4 or 8 bytes)</td>
</tr>
<tr>
<td>q</td>
<td>octal signed short (2 bytes)</td>
</tr>
<tr>
<td>r</td>
<td>whitespace</td>
</tr>
<tr>
<td>s</td>
<td>raw string (variable size)</td>
</tr>
<tr>
<td>t</td>
<td>horizontal tab</td>
</tr>
<tr>
<td>u</td>
<td>decimal unsigned short (2 bytes)</td>
</tr>
<tr>
<td>v</td>
<td>decimal signed int (1 byte)</td>
</tr>
</tbody>
</table>
w  default radix unsigned short (2 bytes)

x  hexadecimal short (2 bytes)

y  decoded time64_t (8 bytes)

The /, \, and ? formatting dcmds can also be used to write to the target’s virtual address space, physical address space, or object file by specifying one of the following modifiers as the first format character, and then specifying a list of words that are either immediate values or expressions enclosed in square brackets preceded by a dollar sign ($ [ ] ).

The write modifiers are:

v  Write the lowest byte of the value of each expression to the target beginning at the location specified by dot.

w  Write the lowest two bytes of the value of each expression to the target beginning at the location specified by dot.

W  Write the lowest 4 bytes of the value of each expression to the target beginning at the location specified by dot.

Z  Write the complete 8 bytes of the value of each expression to the target beginning at the location specified by dot.

The /, \, and ? formatting dcmds can also be used to search for a particular integer value in the target’s virtual address space, physical address space, and object file, respectively, by specifying one of the following modifiers as the first format character, and then specifying a value and optional mask. The value and mask are each specified as either immediate values or expressions enclosed in square brackets preceded by a dollar sign. If only a value is specified, mdb reads integers of the appropriate size and stops at the address containing the matching value. If a value V and mask M are specified, mdb reads integers of the appropriate size and stops at the address containing a value X where \((X \& M) == V\). At the completion of the dcmd, dot is updated to the address containing the match. If no match is found, dot is left at the last address that was read.

The search modifiers are:

l  Search for the specified 2-byte value.

L  Search for the specified 4-byte value.

M  Search for the specified 8-byte value.
Notice that for both user and kernel targets, an address space is typically composed of a set of discontiguous segments. It is not legal to read from an address that does not have a corresponding segment. If a search reaches a segment boundary without finding a match, it will abort when the read past the end of the segment boundary fails.

**Execution Control**

mdb provides facilities for controlling and tracing the execution of a live running program. Currently, only the user process target provides support for execution control. mdb provides a simple model of execution control: a target process can be started from within the debugger using ::run, or mdb can attach to an existing process using :A, ::attach, or the -p command-line option, as described below. A list of traced software events can be specified by the user. Each time a traced event occurs in the target process, all threads in the target stop, the thread that triggered the event is chosen as the representative thread, and control returns to the debugger. Once the target program is set running, control can be asynchronously returned to the debugger by typing the user-defined interrupt character (typically ^C).

A **software event** is a state transition in the target program that is observed by the debugger. For example, the debugger may observe the transition of a program counter register to a value of interest (a breakpoint) or the delivery of a particular signal.

A **software event specifier** is a description of a class of software events that is used by the debugger to instrument the target program in order to observe these events. The ::events dcmd is used to list the software event specifiers. A set of standard properties is associated with each event specifier, as described under ::events, below.

The debugger can observe a variety of different software events, including breakpoints, watchpoints, signals, machine faults, and system calls. New specifiers can be created using ::bp, ::fltbp, ::sigbp, ::sysbp, or ::wp. Each specifier has an associated callback (an mdb command string to execute as if it had been typed at the command prompt) and a set of properties, as described below. Any number of specifiers for the same event may be created, each with different callbacks and properties. The current list of traced events and the properties of the corresponding event specifiers can be displayed using the ::events dcmd. The event specifier properties are defined as part of the description of the ::events and ::evset dcmds, below.

The execution control built-in dcmds, described below, are always available, but will issue an error message indicating they are not supported if applied to a target that does not support execution control. For more information about the interaction of exec, attach, release, and job control with debugger execution control, refer to NOTES, below.

**Event Callbacks**

The ::evset dcmd and event tracing dcmds allow you to associate an event callback (using the -c option) with each event specifier. The event callbacks are strings that represent mdb commands to execute when the corresponding event occurs in the target. These commands are executed as if they had been typed at the command
Prompt. Before executing each callback, the dot variable is set to the value of the representative thread’s program counter and the "hits" variable is set to the number of times this specifier has been matched, including the current match.

If the event callbacks themselves contain one or more commands to continue the target (for example, ::cont or ::step), these commands do not immediately continue the target and wait for it to stop again. Instead, inside of an event callback, the continue dcmds note that a continue operation is now pending, and then return immediately. Therefore, if multiple dcmds are included in an event callback, the step or continue dcmd should be the last command specified. Following the execution of all event callbacks, the target will immediately resume execution if all matching event callbacks requested a continue. If conflicting continue operations are requested, the operation with the highest precedence determines what type of continue will occur. The order of precedence from highest to lowest is: step, step-over (next), step-out, continue.

**Thread Support**

mdb provides facilities to examine the stacks and registers of each thread associated with the target. The persistent "thread" variable contains the current representative thread identifier. The format of the thread identifier depends on the target. The ::regs and ::fpregs dcmds can be used to examine the register set of the representative thread, or of another thread if its register set is currently available. In addition, the register set of the representative thread is exported as a set of named variables. The user can modify the value of one or more registers by applying the > dcmd to the corresponding named variable.

The mdb kernel target exports the virtual address of the corresponding internal thread structure as the identifier for a given thread. The Solaris Modular Debugger Guide provides more information on debugging support for threads in the Solaris kernel. The mdb process target provides proper support for examination of multi-threaded user processes that use the native lwp_* interfaces, /usr/lib/libthread.so or /usr/lib/lwp/libthread.so. When debugging a live user process, mdb will detect if a single threaded process diopens or closes libthread and will automatically adjust its view of the threading model on-the-fly. The process target thread identifiers will correspond to either the lwpid_t, thread_t, or pthread_t of the representative, depending on the threading model used by the application.

If mdb is debugging a user process target and the target makes use of compiler-supported thread-local storage, mdb will automatically evaluate symbol names referring to thread-local storage to the address of the storage corresponding to the current representative thread. The ::tls built-in dcmd can be used to display the value of the symbol for threads other than the representative thread.

**Built-in dcmds**

mdb provides a set of built-in dcmds that are always defined. Some of these dcmds are only applicable to certain targets: if a dcmd is not applicable to the current target, it will fail and print a message indicating "command is not supported by current target". In many cases, mdb provides a mnemonic equivalent (::identifier) for the legacy adb(1) dcmd names. For example, ::quit is provided as the equivalent of $q.

Programmers who are experienced with adb(1) or who appreciate brevity or arcana may prefer the $ or :: forms of the built-ins. Programmers who are new to mdb may...
prefer the more verbose `::` form. The built-ins are shown in alphabetical order. If a `$` or `:` form has a `::identifier` equivalent, it is shown underneath the `::identifier` form. The built-in dcmds are:

```plaintext
> variable-name
>`/modifier/variable-name
   Assign the value of dot to the specified named variable. Some variables are read-only and may not be modified. If the `>` is followed by a modifier character surrounded by `/ /`, then the value is modified as part of the assignment. The modifier characters are:
   c   unsigned char quantity (1-byte)
   s   unsigned short quantity (2-byte)
   i   unsigned int quantity (4-byte)
   l   unsigned long quantity (4-byte in 32-bit, 8-byte in 64-bit)
```

Notice that these operators do not perform a cast. Instead, they fetch the specified number of low-order bytes (on little-endian architectures) or high-order bytes (big-endian architectures). Modifiers are provided for backwards compatibility; the `mdb */modifier/` and `%/modifier/` syntax should be used instead.

```plaintext
$. < macro-name
   Read and execute commands from the specified macro file. The filename may be given as an absolute or relative path. If the filename is a simple name (that is, if it does not contain a `'/'), `mdb` will search for it in the macro file include path. If another macro file is currently being processed, this file is closed and replaced with the new file.

$. < < macro-name
   Read and execute commands from the specified macro file (as with `. <`), but do not close the current open macro file.

$. ?
   Print the process-ID and current signal of the target if it is a user process or core file, and then print the general register set of the representative thread.

[ address ] $. c [ count ]
   Print a C stack backtrace, including stack frame pointer information. If the dcmd is preceded by an explicit `address`, a backtrace beginning at this virtual memory address is displayed. Otherwise the stack of the representative thread is displayed. If an optional count value is given as an argument, no more than `count` arguments are displayed for each stack frame in the output.

[ base ] $. d
   Get or set the default output radix. If the dcmd is preceded by an explicit expression, the default output radix is set to the given `base`; otherwise the current radix is printed in base 10 (decimal). The default radix is base 16 (hexadecimal).
$e
Print a list of all known external (global) symbols of type object or function, the
value of the symbol, and the first 4 (32-bit mdb) or 8 (64-bit mdb) bytes stored at this
location in the target's virtual address space. The ::nm dcmd provides more
flexible options for displaying symbol tables.

$P prompt-string
Set the prompt to the specified prompt-string. The default prompt is ‘>’. The prompt
can also be set using ::set -P or the -P command-line option.

distance $s
Get or set the symbol matching distance for address-to-symbol-name conversions.
The symbol matching distance modes are discussed along with the -s
command-line option under OPTIONS, below. The symbol matching distance may
also be modified using the ::set -s option. If no distance is specified, the current
setting is displayed.

$V
Print a list of the named variables that have non-zero values. The ::vars dcmd
provides other options for listing variables.

width $W
Set the output page width to the specified value. Typically, this command is not
necessary as mdb queries the terminal for its width and handles resize events.

$W
Re-open the target for writing, as if mdb had been executed with the -w option on
the command line. Write mode can also be enabled with the ::set -w option.

[ pid ] ::attach [ core | pid ]

If the user process target is active, attach to and debug the specified process-ID or
core file. The core file pathname should be specified as a string argument. The
process-ID may be specified as the string argument, or as the value of the
expression preceding the dcmd. Recall that the default base is hexadecimal, so
decimal PIDs obtained using pgrvep(1) or ps(1) should be preceded with "0t" when
specified as expressions.

[ address ] ::bp [+| -dDesT] [-c cmd] [-n count] sym ...

Set a breakpoint at the specified locations. The ::bp dcmd sets a breakpoint at each
address or symbol specified, including an optional address specified by an explicit
expression preceding the dcmd, and each string or immediate value following the
dcmd. The arguments may either be symbol names or immediate values denoting a
particular virtual address of interest. If a symbol name is specified, it may refer to a
symbol that cannot yet be evaluated in the target process. That is, it may consist of
an object name and function name in a load object that has not yet been opened. In
this case, the breakpoint is deferred and it will not be active in the target until an
object matching the given name is loaded. The breakpoint will be automatically
enabled when the load object is opened. Breakpoints on symbols defined in a
shared library should always be set using a symbol name and not using an address
expression, as the address may refer to the corresponding Procedure Linkage Table (PLT) entry instead of the actual symbol definition. Breakpoints set on PLT entries may be overwritten by the run-time link-editor when the PLT entry is subsequently resolved to the actual symbol definition. The -d, -D, -e, -s, -t, -T, -c, and -n options have the same meaning as they do for the ::evset dcmd, as described below. If the :b form of the dcmd is used, a breakpoint is only set at the virtual address specified by the expression preceding the dcmd. The arguments following the :b dcmd are concatenated together to form the callback string. If this string contains meta-characters, it must be quoted.

::cat filename ...
Concatenate and display files. Each filename may specified as a relative or absolute pathname. The file contents will be printed to standard output, but will not be passed to the output pager. This dcmd is intended to be used with the | operator; the programmer can initiate a pipeline using a list of addresses stored in an external file.

::cont [ SIG ]
:c [ SIG ]
Suspend the debugger, continue the target program, and wait for it to terminate or stop following a software event of interest. If the target is already running because the debugger was attached to a running program with the -nostop option enabled, this dcmd simply waits for the target to terminate or stop after an event of interest. If an optional signal name or number (see signal(3HEAD)) is specified as an argument, the signal is immediately delivered to the target as part of resuming its execution. If the SIGINT signal is traced, control may be asynchronously returned to the debugger by typing the user-defined interrupt character (usually ^C). This SIGINT signal will be automatically cleared and will not be observed by the target the next time it is continued. If no target program is currently running, ::cont will start a new program running as if by ::run.

address ::context
address $p
Context switch to the specified process. A context switch operation is only valid when using the kernel target. The process context is specified using the address of its proc structure in the kernel’s virtual address space. The special context address "0" is used to denote the context of the kernel itself. mdb can only perform a context switch when examining a crash dump if the dump contains the physical memory pages of the specified user process (as opposed to just kernel pages). The kernel crash dump facility can be configured to dump all pages or the pages of the current user process using dumpadm(1M). The ::status dcmd can be used to display the contents of the current crash dump.

When the user requests a context switch from the kernel target, mdb constructs a new target representing the specified user process. Once the switch occurs, the new target interposes its dcmds at the global level: thus the / dcmd will now format and display data from the virtual address space of the user process, the ::mappings dcmd will display the mappings in the address space of the user process, and so on. The kernel target can be restored by executing 0::context.
::dcmds
List the available dcmds and print a brief description for each one.

[address] ::delete [id | all]
[address] ::d [id | all]
Delete the event specifiers with the given id number. The id number argument is interpreted in decimal by default. If an optional address is specified preceding the dcmd, all event specifiers that are associated with the given virtual address are deleted (for example, all breakpoints or watchpoints affecting that address). If the special argument "all" is given, all event specifiers are deleted, except those that are marked sticky (T flag). The ::events dcmd displays the current list of event specifiers.

[address] ::dis [ -fw | -n count ] [address]
Disassemble starting at or around the address specified by the final argument, or the current value of dot. If the address matches the start of a known function, the entire function is disassembled. Otherwise, a "window" of instructions before and after the specified address is printed in order to provide context. By default, instructions are read from the target’s virtual address space. If the -f option is present, instructions are read from the target’s object file instead. The -f option is enabled by default if the debugger is not currently attached to a live process, core file, or crash dump. The -w option can be used to force "window"-mode, even if the address is the start of a known function. The size of the window defaults to ten instructions; the number of instructions can be specified explicitly using the -n option.

::disasms
List the available disassembler modes. When a target is initialized, mdb will attempt to select the appropriate disassembler mode. The user can change the mode to any of the modes listed using the ::dismode dcmd.

::dismode [mode]
Set [mode]
Get or set the disassembler mode. If no argument is specified, print the current disassembler mode. If a mode argument is specified, switch the disassembler to the specified mode. The list of available disassemblers can be displayed using the ::disasms dcmd.

::dmods [-l] [module-name]
List the loaded debugger modules. If the -l option is specified, the list of the dcmds and walkers associated with each dmod is printed below its name. The output can be restricted to a particular dmod by specifying its name as an additional argument.

[address] ::dump [-egrstu] [-f | -p] [-g bytes] [-w paragraphs]
Print a hexadecimal and ASCII memory dump of the 16-byte aligned region of memory containing the address specified by dot. If a repeat count is specified for ::dump, this is interpreted as a number of bytes to dump rather than a number of iterations. The ::dump dcmd also recognizes the following options:
mdb(1)

- `e` Adjusts for endian-ness. The `-e` option assumes 4-byte words. The `-f` option can be used to change the default word size.

- `f` Reads data from the object file location corresponding to the given virtual address instead of from the target's virtual address space. The `-f` option is enabled by default if the debugger is not currently attached to a live process, core file, or crash dump.

- `g bytes` Displays bytes in groups of `bytes`. The default group size is 4 bytes. The group size must be a power of two that divides the line width.

- `p` Interprets `address` as a physical address location in the target's address space instead of a virtual address.

- `q` Does not print an ASCII decoding of the data.

- `r` Numbers lines relative to the start address instead of with the explicit address of each line. This option implies the `-u` option.

- `s` Elides repeated lines.

- `t` Only reads from and displays the contents of the specified addresses, instead of reading and printing entire lines.

- `u` Unaligns output instead of aligning the output at a paragraph boundary.

- `w paragraphs` Displays paragraphs at 16-byte paragraphs per line. The default number of paragraphs is one. The maximum value accepted for `-w` is 16.

::`echo` [string | value ...]
Print the arguments separated by blanks and terminated by a NEWLINE to standard output. Expressions enclosed in `$[ ]` will be evaluated to a value and printed in the default base.

::`eval` command
Evaluate and execute the specified string as a command. If the command contains metacharacters or whitespace, it should be enclosed in double or single quotes.

::`events` [-av ]

$::b [-av ]
Display the list of software event specifiers. Each event specifier is assigned a unique ID number that can be used to delete or modify it at a later time. The debugger may also have its own internal events enabled for tracing. These events will only be displayed if the `-a` option is present. If the `-v` option is present, a more verbose display, including the reason for any specifier inactivity, will be shown. Here is some sample output:

> ::events
  ID $ TA HT LM Description Action
  ------ -- -- -- -------------- --------
The following table explains the meaning of each column. A summary of this information is available using `::help events`.

**ID**
The event specifier identifier. The identifier will be shown in square brackets [ ] if the specifier is enabled, in parentheses ( ) if the specifier is disabled, or in angle brackets <> if the target program is currently stopped on an event that matches the given specifier.

**S**
The event specifier state. The state will be one of the following symbols:

- The event specifier is idle. When no target program is running, all specifiers are idle. When the target program is running, a specifier may be idle if it cannot be evaluated (for example, a deferred breakpoint in a shared object that is not yet loaded).

+ The event specifier is active. When the target is continued, events of this type will be detected by the debugger.

* The event specifier is armed. This state means that the target is currently running with instrumentation for this type of event. This state is only visible if the debugger is attached to a running program with the `-onostop` option.

! The event specifier was not armed due to an operating system error. The `::events -v` option can be used to display more information about the reason the instrumentation failed.

**TA**
The Temporary, Sticky, and Automatic event specifier properties. One or more of the following symbols may be shown:

τ The event specifier is temporary, and will be deleted the next time the target stops, regardless of whether it is matched.
The event specifier is sticky, and will be not be deleted by ::delete all or z. The specifier can be deleted by explicitly specifying its id number to ::delete.

The event specifier will be automatically disabled when the hit count is equal to the hit limit.

The event specifier will be automatically deleted when the hit count is equal to the hit limit.

The target will automatically stop when the hit count is equal to the hit limit.

<table>
<thead>
<tr>
<th>HT</th>
<th>The current hit count. This column displays the number of times the corresponding software event has occurred in the target since the creation of this event specifier.</th>
</tr>
</thead>
<tbody>
<tr>
<td>LM</td>
<td>The current hit limit. This column displays the limit on the hit count at which the auto-disable, auto-delete, or auto-stop behavior will take effect. These behaviors can be configured using the ::evset dcmd, described below.</td>
</tr>
</tbody>
</table>

**Description**

A description of the type of software event that is matched by the given specifier.

**Action**

The callback string to execute when the corresponding software event occurs. This callback is executed as if it had been typed at the command prompt.

```
[id] ::evset [+/dDestT] [-c cmd] [-n count] id ...
```

Modify the properties of one or more software event specifiers. The properties are set for each specifier identified by the optional expression preceding the dcmd and an optional list of arguments following the dcmd. The argument list is interpreted as a list of decimal integers, unless an explicit radix is specified. The ::evset dcmd recognizes the following options:

- **-d** Enables the event specifier when the hit count reaches the hit limit. If the +d form of the option is given, this behavior is disabled. Once an event specifier is disabled, the debugger will remove any corresponding instrumentation and will ignore the corresponding software events until the specifier is subsequently re-enabled. If the -n option is not present, the specifier is disabled immediately.

- **-D** Deletes the event specifier when the hit count reaches the hit limit. If the +D form of the option is given, this behavior is disabled. The -D option takes precedence over the -d option. The hit limit can be configured using the -n option.

- **-e** Enables the event specifier. If the +e form of the option is given, the specifier is disabled.
-s  Stops the target program when the hit count reaches the hit limit. If the +s form of the option is given, this behavior is disabled. The -s behavior tells the debugger to act as if the ::cont were issued following each execution of the specifier’s callback, except for the Nth execution, where N is the current value of the specifier’s hit limit. The -s option takes precedence over both the -D option and the -d option.

-t  Marks the event specifier as temporary. Temporary specifiers are automatically deleted the next time the target stops, regardless of whether it stopped as the result of a software event corresponding to the given specifier. If the +t form of the option is given, the temporary marker is removed. The -t option takes precedence over the -T option.

-T  Marks the event specifier as sticky. Sticky specifiers will not be deleted by ::delete all or :z. They can be deleted by specifying the corresponding specifier ID as an explicit argument to ::delete. If the +T form of the option is given, the sticky property is removed. The default set of event specifiers are all initially marked sticky.

-c  Executes the specified cmd string each time the corresponding software event occurs in the target program. The current callback string can be displayed using ::events.

-n  Sets the current value of the hit limit to count. If no hit limit is currently set and the -n option does not accompany -s or D, the hit limit will be set to one.

A summary of this information is available using ::help evset.

::files
$f
Print a list of the known source files (symbols of type STT_FILE present in the various target symbol tables).

[flt] ::fltbp [+/ -dDestT] [-c cmd] [-n count] fli ... Trace the specified machine faults. The faults are identified using an optional fault number preceding the dcmd, or a list of fault names or numbers (see <sys/fault.h>) following the dcmd. The -d, -D, -e, -s, -t, -T, -c, and -n options have the same meaning as they do for the ::evset dcmd.

[thread] ::fpregs
[thread ] $x, $X, $y, $Y
Print the floating-point register set of the representative thread. If a thread is specified, the floating point registers of that thread are displayed. The thread expression should be one of the thread identifiers described under Thread Support, above.

::formats
List the available output format characters for use with the /, \, ?, and = formatting dcmds. The formats and their use is described under Formatting dcmds, above.
::grep command
Evaluate the specified command string, and then print the old value of dot if the
new value of dot is non-zero. If the command contains whitespace or
metacharacters, it must be quoted. The ::grep dcmd can be used in pipelines to
filter a list of addresses.

::help [ dcmd-name ]
With no arguments, the ::help dcmd prints a brief overview of the help facilities
available in mdb. If a dcmd-name is specified, mdb will print a usage summary for
that dcmd.

signal :i
If the target is a live user process, ignore the specified signal and allow it to be
delivered transparently to the target. All event specifiers that are tracing delivery of
the specified signal will be deleted from the list of traced events. By default, the set
of ignored signals is initialized to the complement of the set of signals that cause a
process to dump core by default (see signal(3HEAD)), except for SIGINT, which
is traced by default.

$:
Display the list of signals that are ignored by the debugger and that will be handled
directly by the target. More information on traced signals can be obtained using the
::events dcmd.

::kill
: k
Forcibly terminate the target if it is a live user process. The target will also be
dforcibly terminated when the debugger exits if it was created by the debugger using
::run.

$:
Print the LWPID of the representative thread, if the target is a user process.

$:
Print the LWPIDs of each LWP in the target, if the target is a user process.

[ address ] ::list type member [ variable-name ]
Walk through the elements of a linked list data structure and print the address of
each element in the list. The address of the first element in the list can be specified
using an optional address. Otherwise, the list is assumed to start at the current
value of dot. The type parameter must name a C struct or union type and is used to
describe the type of the list elements so that mdb can read in objects of the
appropriate size. The member parameter is used to name the member of type that
contains a pointer to the next list element. The ::list dcmd will continue iterating
until a NULL pointer is encountered, the first element is reached again (a circular
list), or an error occurs while reading an element. If the optional variable-name is
specified, the specified variable will be assigned the value returned at each step of
the walk when mdb invokes the next stage of a pipeline. The ::list dcmd may
only be used with objects that contain symbolic debugging information designed
for use with mdb. Refer to NOTES, Symbolic Debugging Information, below
for more information.
::load [-s] module-name
Load the specified dmod. The module name may be given as an absolute or relative path. If module-name is a simple name (that is, does not contain a '/'), mdb will search for it in the module library path. Modules with conflicting names may not be loaded; the existing module must be unloaded first. If the -s option is present, mdb will remain silent and not issue any error messages if the module is not found or could not be loaded.

::log [-d | -e] filename
$> [filename]
Enable or disable the output log. mdb provides an interactive logging facility where both the input commands and standard output can be logged to a file while still interacting with the user. The -e option enables logging to the specified file, or re-enables logging to the previous log file if no filename is given. The -d option disables logging. If the $> dcmd is used, logging is enabled if a filename argument is specified; otherwise, logging is disabled. If the specified log file already exists, mdb appends any new log output to the file.

::map command
Map the value of dot to a corresponding value using the command specified as a string argument, and then print the new value of dot. If the command contains whitespace or metacharacters, it must be quoted. The ::map dcmd can be used in pipelines to transform the list of addresses into a new list of addresses.

[address] ::mappings [name]
[address] $m [name]
Print a list of each mapping in the target’s virtual address space, including the address, size, and description of each mapping. If the dcmd is preceded by an address, mdb will only show the mapping that contains the given address. If a string name argument is given, mdb will only show the mapping matching that description.

::next [SIG]
:e [SIG]
Step the target program one instruction, but step over subroutine calls. If an optional signal name or number (see signal(3HEAD)) is specified as an argument, the signal is immediately delivered to the target as part of resuming its execution. If no target program is currently running, ::next will start a new program running as if by ::run and stop at the first instruction.

[address] ::nm [ -DPdghnopuvx ] [ -t types ]
[ -f format ] [ object ]
Print the symbol tables associated with the current target. If an optional address preceding the dcmd is specified, only the symbol table entry for the symbol corresponding to address is displayed. If an object is specified, only the symbol table for this load object is displayed. The ::nm dcmd also recognizes the following options:

-D
Prints .dynsym (dynamic symbol table) instead of .symtab.
-P  Prints the private symbol table instead of .symtab.
-d  Prints value and size fields in decimal.
-g  Prints only global symbols.
-h  Suppresses the header line.
-n  Sorts symbols by name.
-o  Prints value and size fields in octal.
-p  Prints symbols as a series of :nmadd commands. This option can be used with -P to produce a macro file that can be subsequently read into the debugger with $<.
-u  Prints only undefined symbols.
-v  Sorts symbols by value.
-x  Prints value and size fields in hexadecimal.
-t type[type ... ]  Prints only symbols of the specified type(s). The valid type argument strings are:
   noty   STT_NOTYPE
   objt   STT_OBJECT
   func   STT_FUNC
   sect   STT_SECTION
   file   STT_FILE
   comm   STT_COMMON
   tls    STT_TLS
   regi   STT_SPARC_REGISTER
-f format[format ... ]  Prints only the specified symbol information. The valid format argument strings are:
   ndx    symbol table index
   val    symbol value
   size   size in bytes
   type   symbol type
   bind   binding
   oth    other
   shndx  section index
   name   symbol name
Add the specified symbol name to the private symbol table. *mdb* provides a private, configurable symbol table that can be used to interpose on the target’s symbol table, as described under *Symbol Name Resolution* above. The ::nmadd dcmd also recognizes the following options:

- **-e**  Sets the size of the symbol to end - value.
- **-f**  Sets the type of the symbol to STT_FUNC.
- **-o**  Sets the type of the symbol to STT_OBJECT.
- **-s**  Sets the size of the symbol to size.

::nmdel name
Delete the specified symbol name from the private symbol table.

::objects
Print a map of the target’s virtual address space, showing only those mappings that correspond to the primary mapping (usually the text section) of each of the known load objects.

::offsetof type member
Print the offset of the specified member of the specified type. The type should be the name of a C structure. The offset is printed in bytes, unless the member is a bit-field, in which case the offset may be printed in bits. The output is always suffixed with the appropriate units for clarity. The type name may use the backquote (‘) scoping operator described under *Symbol Name Resolution*, above. The ::offsetof dcmd may only be used with objects that contain symbolic debugging information designed for use with *mdb*. Refer to NOTES, *Symbolic Debugging Information*, below for more information.

::print address [-aCdLptx] [-c lim] [-l lim] [type [member ...]]
Print the data structure at the specified virtual address using the given type information. The type parameter may name a C struct, union, enum, fundamental integer type, or a pointer to any of these types. If the type name contains whitespace (for example, "struct foo"), it must be enclosed in single or double quotes. The type name may use the backquote (‘) scoping operator described under *Symbol Name Resolution*, above. If the type is a structured type, the ::print dcmd will recursively print each member of the struct or union. If the type argument is not present and a static or global STT_OBJECT symbol matches the address, ::print will infer the appropriate type automatically. If the type argument is specified, it may be followed by an optional list of member expressions, in which case only those members and submembers of the specified type are displayed. If type contains other structured types, each member string may refer to a sub-structure element by forming a list of member names separated by period (‘.’) delimiters. The ::print dcmd may only be used with objects that contain symbolic debugging information designed for use with *mdb*. Refer to NOTES,
Symbolic Debugging Information, below for more information. After
displaying the data structure, ::print increments dot by the size of type in bytes.

If the -a option is present, the address of each member is displayed. If the -p
option is present, ::print interprets address as a physical memory address instead
of a virtual memory address. If the -t option is present, the type of each member is
displayed. If the -d or -x options are present, all integers are displayed in decimal
(-d) or hexadecimal (-x). By default, a heuristic is used to determine if the value
should be displayed in decimal or hexadecimal. The number of characters in a
character array that will be read and displayed as a string can be limited with the
-c option. If the -C option is present, no limit is enforced. The number of elements
in a standard array that will be read and displayed can be limited with the -l
option. If the -L option is present, no limit is enforced and all array elements are
shown. The default values for -c and -l can be modified using ::set or the -o
command-line option as described under OPTIONS, below.

::quit
$q
Quit the debugger.

[thread] ::regs
[thread] $r
Print the general purpose register set of the representative thread. If a thread is
specified, the general purpose register set of that thread is displayed. The thread
expression should be one of the thread identifiers described under Thread
Support, above.

::release [-a]
::R [-a]
Release the previously attached process or core file. If the -a option is present, the
process is released and left stopped and abandoned. It can subsequently be
continued by prun(1) (see proc(1)) or it can be resumed by applying mdb or
another debugger. By default, a released process is forcibly terminated if it was
created by mdb using ::run, or it is released and set running if it was attached to
by mdb using the -p option or using the ::attach or :A dcmds.

::run [args ...]
::r [args ...]
Start a new target program running with the specified arguments and attach to it.
The arguments are not interpreted by the shell. If the debugger is already
examining a live running program, it will first detach from this program as if by
::release.

::set [-WF] [+/-o option] [-s distance] [-I path]
[-L path] [-P prompt]
Get or set miscellaneous debugger properties. If no options are specified, the
current set of debugger properties is displayed. The ::set dcmd recognizes the
following options:
- F Forcibly takes over the next user process that ::attach is applied to, as
  if mdb had been executed with the -F option on the command line.
Sets the default path for locating macro files. The path argument may contain any of the special tokens described for the \-I command-line option under OPTIONS below.

-\L
Sets the default path for locating debugger modules. The path argument may contain any of the special tokens described for the \-I command-line option under OPTIONS below.

-\o
Enables the specified debugger option. If the \+\o form is used, the option is disabled. The option strings are described along with the \-\o command-line option under OPTIONS below.

-\P
Sets the command prompt to the specified prompt string.

-\s
Sets the symbol matching distance to the specified distance. Refer to the description of the \-s command-line option under OPTIONS below for more information.

-\w
Re-opens the target for writing, as if mdb had been executed with the \-w option on the command line.

\[signal\] ::\sigbp [\+/\-\d\DestT] [-c cmd] [\-n count] SIG ...
\[signal\] ::\st [\+/\-\d\DestT] [-c cmd] [\-n count] SIG ...
Trace delivery of the specified signals. The signals are identified using an optional signal number preceding the dcmd, or a list of signal names or numbers (see signal(3HEAD)) following the dcmd. The \-d, \-D, \-e, \-s, \-t, \-T, \-c, and \-n options have the same meaning as they do for the ::\evset dcmd. Initially, the set of signals that cause the process to dump core by default (see signal(3HEAD)) and SIGINT are traced.

::\sizeof type
Print the size of the specified type in bytes. The type parameter may name a C struct, union, enum, fundamental integer type, or a pointer to any of these types. The type name may use the backquote (\') scoping operator described under Symbol Name Resolution, above. The ::\sizeof dcmd may only be used with objects that contain symbolic debugging information designed for use with mdb. Refer to NOTES, Symbolic Debugging Information, below for more information.

\[address\] ::\s\tack [count ]
\[address\] ::\s\tack \$c [count ]
Print a C stack backtrace. If the dcmd is preceded by an explicit address, a backtrace beginning at this virtual memory address is displayed. Otherwise the stack of the representative thread is displayed. If an optional count value is given as an argument, no more than count arguments are displayed for each stack frame in the output.

::\status
Print a summary of information related to the current target.

::\step [ over | out ] [ SIG ]
::\s [ SIG ]
::\u [ SIG ]
Step the target program one instruction. If an optional signal name or number (see `signal(3HEAD)) is specified as an argument, the signal is immediately delivered to the target as part of resuming its execution. If the optional "over" argument is specified, `::step` will step over subroutine calls. The `::step` over argument is the same as the `::next` dcmd. If the optional "out" argument is specified, the target program will continue until the representative thread returns from the current function. If no target program is currently running, `::stepout` will start a new program running as if by `::run` and stop at the first instruction. The `::s` dcmd is the same as `::step`. The `::u` dcmd is the same as `::step`.

```
[ syscall ] ::sysbp [ +/-dDestT ] [ -io ] [ -c cmd ]
[ -n count ] syscall...
```

Trace entry to or exit from the specified system calls. The system calls are identified using an optional system call number preceding the dcmd, or a list of system call names or numbers (see `<sys/syscall.h>`) following the dcmd. If the `-i` option is specified (the default), the event specifiers trigger on entry into the kernel for each system call. If the `-o` option is specified, the event specifiers trigger on exit out from the kernel. The `-d`, `-D`, `-e`, `-s`, `-t`, `-T`, `-c`, and `-n` options have the same meaning as they do for the `::evset` dcmd.

```
thread ::tls symbol
```

Print the address of the storage for the specified thread-local storage (TLS) symbol in the context of the specified thread. The thread expression should be one of the thread identifiers described under Thread Support, above. The symbol name may use any of the scoping operators described under Symbol Name Resolution, above.

```
::typeset [ +/-t] variable-name ...
```

Set attributes for named variables. If one or more variable names are specified, they are defined and set to the value of dot. If the `-t` option is present, the user-defined tag associated with each variable is set. If the `+t` option is present, the tag is cleared. If no variable names are specified, the list of variables and their values is printed.

```
::unload module-name
```

Unload the specified dmod. The list of active dmods may be printed using the `::dmods` dcmd. Built-in modules may not be unloaded. Modules that are busy (that is, provide dcmds that are currently executing) may not be unloaded.

```
::unset variable-name ...
```

Unset (remove) the specified variable(s) from the list of defined variables. Some variables exported by `mdb` are marked as persistent, and may not be unset by the user.

```
::vars [ -npt]
```

Print a listing of named variables. If the `-n` option is present, the output is restricted to variables that currently have non-zero values. If the `-p` option is present, the variables are printed in a form suitable for re-processing by the debugger using the `$<` dcmd. This option can be used to record the variables to a
macro file and then restore these values later. If the -t option is present, only the
tagged variables are printed. Variables can be tagged using the -t option of the
::typeset dcmd.

::version
Print the debugger version number.

**address** ::vtop [-a as]
Print the physical address mapping for the specified virtual address, if possible.
The ::vtop dcmd is only available when examining a kernel target, or when
examining a user process inside a kernel crash dump (after a ::context dcmd has
been issued).

When examining a kernel target from the kernel context, the -a option can be used
to specify the address (as) of an alternate address space structure that should be
used for the virtual to physical translation. By default, the kernel’s address space is
used for translation. This option is available for active address spaces even when
the dump content only contains kernel pages.

[ address ] ::walk walker-name [ variable-name ]
Walk through the elements of a data structure using the specified walker. The
available walkers can be listed using the ::walkers dcmd. Some walkers operate
on a global data structure and do not require a starting address. For example, walk
the list of proc structures in the kernel. Other walkers operate on a specific data
structure whose address must be specified explicitly. For example, given a pointer
to an address space, walk the list of segments. When used interactively, the ::walk
dcmd will print the address of each element of the data structure in the default
base. The dcmd can also be used to provide a list of addresses for a pipeline. The
walker name may use the backquote (‘) scoping operator described under dcmd
and Walker Name Resolution, above. If the optional variable-name is specified,
the specified variable will be assigned the value returned at each step of the walk
when mdb invokes the next stage of the pipeline.

::walkers
List the available walkers and print a brief description for each one.

::whence [ -v ] name ...
::which [ -v ] name ...
Print the dmod that exports the specified dcmds and walkers. These dcmds can be
used to determine which dmod is currently providing the global definition of the
given dcmd or walker. Refer to the section on dcmd and Walker Name
Resolution above for more information on global name resolution. The -v option
will cause the dcmd to print the alternate definitions of each dcmd and walker in
order of precedence.

addr [,len] ::wp [ +/ -dDestT ] [ -rwx ] [ -c cmd ] [ -n count ]
addr [,len] :a [ cmd ... ]
addr [,len] :p [ cmd ... ]
addr [,len] :w [ cmd ... ]
Set a watchpoint at the specified address. The length in bytes of the watched region
may be set by specifying an optional repeat count preceding the dcmd. If no length
is explicitly set, the default is one byte. The `::wp` dcmd allows the watchpoint to be configured to trigger on any combination of read (-r option), write (-w option), or execute (-x option) access. The -d, -D, -e, -s, -t, -T, -c, and -n options have the same meaning as they do for the `::evset` dcmd. The `:a` dcmd sets a read access watchpoint at the specified address. The `:p` dcmd sets an execute access watchpoint at the specified address. The `:w` dcmd sets a write access watchpoint at the specified address. The arguments following the `:a`, `:p`, and `:w` dc cmds are concatenated together to form the callback string. If this string contains meta-characters, it must be quoted.

`::xdata`
List the external data buffers exported by the current target. External data buffers represent information associated with the target that cannot be accessed through standard target facilities (that is, an address space, symbol table, or register set). These buffers may be consumed by dcmds; for more information, refer to the Solaris Modular Debugger Guide.

`:z`  
Delete all event specifiers from the list of traced software events. Event specifiers can also be deleted using `::delete`.

**OPTIONS**  
The following options are supported:

- **-A**  
  Disables automatic loading of `mdb` modules. By default, `mdb` attempts to load debugger modules corresponding to the active shared libraries in a user process or core file, or to the loaded kernel modules in the live operating system or an operating system crash dump.

- **-f**  
  Forces raw file debugging mode. By default, `mdb` attempts to infer whether the object and core file operands refer to a user executable and core dump or to a pair of operating system crash dump files. If the file type cannot be inferred, the debugger will default to examining the files as plain binary data. The `-f` option forces `mdb` to interpret the arguments as a set of raw files to examine.

- **-F**  
  Forcibly takes over the specified user process, if necessary. By default, `mdb` refuses to attach to a user process that is already under the control of another debugging tool, such as `truss(1)`. With the `-F` option, `mdb` attaches to these processes anyway. This may produce unexpected interactions between `mdb` and the other tools attempting to control the process.

- **-I path**  
  Sets default path for locating macro files. Macro files are read using the `$<` or `$<<` dcmds. The path is a sequence of directory names delimited by colon (`:`) characters. The `-I include path` and `-L library path` (see below) may also contain any of the following tokens:

  - **%i**  
    Expands to the current instruction set architecture (ISA) name (‘sparc’, ‘sparcv9’, or ‘i386’).
%o Expands to the old value of the path being modified. This is useful for appending or prepending directories to an existing path.

%p Expands to the current platform string (either `uname -i` or the platform string stored in the process core file or crash dump).

%r Expands to the pathname of the root directory. An alternate root directory may be specified using the `-R` option. If no `-R` option is present, the root directory is derived dynamically from the path to the `mdb` executable itself. For example, if `/bin/mdb` is executed, the root directory will be `/`. If `/net/hostname/bin/mdb` were executed, the root directory would be derived as `/net/hostname`.

%t Expands to the name of the current target. This will either be the literal string `proc` (a user process or user process core file), `kvm` (a kernel crash dump or the live operating system), or `raw` (a raw file).

The default include path for 32-bit `mdb` is:

```
%r/usr/platform/%p/lib/adb:%r/usr/lib/adb
```

The default include path for 64-bit `mdb` is:

```
%r/usr/platform/%p/lib/adb/%i:%r/usr/lib/adb/%i
```

-k Forces kernel debugging mode. By default, `mdb` attempts to infer whether the object and core file operands refer to a user executable and core dump, or to a pair of operating system crash dump files. The `-k` option forces `mdb` to assume these files are operating system crash dump files. If no object or core operand is specified, but the `-k` option is specified, `mdb` defaults to an object file of `/dev/ksyms` and a core file of `/dev/kmem`. Access to `/dev/kmem` is restricted to group sys.

-L path Sets default path for locating debugger modules. Modules are loaded automatically on startup or using the `::load` dcmd. The path is a sequence of directory names delimited by colon (`:`) characters. The `-L` library path may also contain any of the tokens shown for `-I` above.

-m Disables demand-loading of kernel module symbols. By default, `mdb` processes the list of loaded kernel modules and performs demand loading of per-module symbol tables. If the `-m` option is specified, `mdb` will not attempt to process the kernel module list or

---

User Commands 895
provide per-module symbol tables. As a result, mdb modules corresponding to active kernel modules will not be loaded on startup.

-\texttt{M} \hfill Preloads all kernel module symbols. By default, mdb performs demand-loading for kernel module symbols: the complete symbol table for a module is read when an address is that module’s text or data section is referenced. With the -\texttt{M} option, mdb loads the complete symbol table of all kernel modules during startup.

-\texttt{o} \textit{option} \hfill Enables the specified debugger option. If the +\textit{o} form of the option is used, the specified \textit{option} is disabled. Unless noted below, each option is off by default. mdb recognizes the following \textit{option} arguments:

\texttt{adb} \hfill Enables stricter adb(1) compatibility. The prompt will be set to the empty string and many mdb features, such as the output pager, will be disabled.

\texttt{array\_mem\_limit=} \texttt{limit} \hfill Sets the default limit on the number of array members that ::\texttt{print} will display. If \textit{limit} is the special token \texttt{none}, all array members will be displayed by default.

\texttt{array\_str\_limit=} \texttt{limit} \hfill Sets the default limit on the number of characters that ::\texttt{print} will attempt to display as an ASCII string when printing a char array. If \textit{limit} is the special token \texttt{none}, the entire char array will be displayed as a string by default.

\texttt{follow\_exec\_mode=} \texttt{mode} \hfill Sets the debugger behavior for following an exec(2) system call. The \textit{mode} should be one of the following named constants:

\texttt{ask} \hfill If stdout is a terminal device, the debugger stops after the exec(2) system call has returned and then prompts the user to decide whether to follow the exec or stop. If stdout is not a terminal device, the ask mode defaults to \texttt{stop}.

\texttt{follow} \hfill The debugger follows the exec by automatically continuing the target process and resetting all of its mappings and symbol tables based on the new executable. The follow behavior is discussed in more detail under NOTES, Interaction with Exec, below.

\texttt{stop} \hfill The debugger stops following return from the exec system call. The stop behavior is discussed in more detail under NOTES, Interaction with Exec, below.
follow_fork_mode

Sets the debugger behavior for following a fork(2), fork1(2), or vfork(2) system call. The mode should be one of the following named constants:

- **ask**: If stdout is a terminal device, the debugger stops after the fork(2) system call has returned and then prompts the user to decide whether to follow the parent or child. If stdout is not a terminal device, the ask mode defaults to parent.
- **parent**: The debugger follows the parent process, and detaches from the child process and sets it running.
- **child**: The debugger follows the child process, and detaches from the parent process and sets it running.

**ignoreeof**

The debugger will not exit when an EOF sequence (^D) is entered at the terminal. The ::quit dcmd must be used to quit.

**nostop**

Does not stop a user process when attaching to it when the -p option is specified or when the ::attach or :A dcmds are applied. The nostop behavior is described in more detail under NOTES, Process Attach and Release, below.

**pager**

Enables the output pager (default).

**repeatlast**

If a NEWLINE is entered as the complete command at the terminal, mdb repeats the previous command with the current value of dot. This option is implied by -o adb.

**showlmid**

mdb provides support for symbol naming and identification in user applications that make use of link maps other than LM_ID_BASE and LM_ID_LDSO, as described in Symbol Name Resolution, above. Symbols on link maps other than LM_ID_BASE or LM_ID_LDSO will be shown as LMlmid‘library’symbol, where lmid is the link-map ID in the default output radix (16). The user may optionally configure mdb to show the link-map ID scope of all symbols and objects, including those associated with LM_ID_BASE and LM_ID_LDSO, by enabling the showlmid option. Built-in dcmds that deal with object file names will display link-map IDs according to the value of showlmid above, including ::nm, ::mappings, $m, and ::objects.
mdb(1)

- **p pid**  
  Attaches to and stops the specified process-id. `mdb` will use the `/proc/pid/object/a.out` file as the executable file pathname.

- **p prompt**  
  Sets the command prompt. The default prompt is ‘>’.

- **R root**  
  Sets root directory for pathname expansion. By default, the root directory is derived from the pathname of the `mdb` executable itself. The root directory is substituted in place of the `%r` token during pathname expansion.

- **s distance**  
  Sets the symbol matching distance for address-to-symbol-name conversions to the specified `distance`. By default, `mdb` sets the distance to zero, which enables a smart-matching mode. Each ELF symbol table entry includes a value `V` and size `S`, representing the size of the function or data object in bytes. In smart mode, `mdb` matches an address `A` with the given symbol if `A` is in the range `[V, V + S]`. If any non-zero distance is specified, the same algorithm is used, but `S` in the expression above is always the specified absolute distance and the symbol size is ignored.

- **S**  
  Suppresses processing of the user’s `~/.mdbrc` file. By default, `mdb` reads and processes the macro file `.mdbrc` if one is present in the user’s home directory, as defined by `$HOME`. If the `-S` option is present, this file will not be read.

- **u**  
  Forces user debugging mode. By default, `mdb` attempts to infer whether the object and core file operands refer to a user executable and core dump, or to a pair of operating system crash dump files. The `-u` option forces `mdb` to assume these files are not operating system crash dump files.

- **V version**  
  Sets disassembler version. By default, `mdb` attempts to infer the appropriate disassembler version for the debug target. The disassembler can be set explicitly using the `-V` option. The `::disasms` dcmd lists the available disassembler versions.

- **w**  
  Opens the specified object and core files for writing.

- **y**  
  Sends explicit terminal initialization sequences for tty mode. Some terminals, such as `cmdtool(1)`, require explicit initialization sequences to switch into a tty mode. Without this initialization sequence, terminal features such as standout mode may not be available to `mdb`.

**OPERANDS**

The following operands are supported:

- **object**  
  Specifies an ELF format object file to examine. `mdb` provides the ability to examine and edit ELF format executables (`ET_EXEC`), ELF dynamic library files (`ET_DYN`), ELF relocatable object files (`ET_REL`), and operating system `unix.X` symbol table files.
core  Specifies an ELF process core file (ET_CORE), or an operating system crash dump vmcore.X file. If an ELF core file operand is provided without a corresponding object file, mdb will attempt to infer the name of the executable file that produced the core using several different algorithms. If no executable is found, mdb will still execute, but some symbol information may be unavailable.

suffix  Specifies the numerical suffix representing a pair of operating system crash dump files. For example, if the suffix is '3', mdb infers that it should examine the files 'unix.3' and 'vmcore.3'. The string of digits will not be interpreted as a suffix if an actual file of the same name is present in the current directory.

EXIT STATUS  The following exit values are returned:

0  Debugger completed execution successfully.
1  A fatal error occurred.
2  Invalid command line options were specified.

ENVIRONMENT VARIABLES  This variable is used to determine the maximum length of the command history list. If this variable is not present, the default length is 128.

HISTSIZE  This variable is used to determine the pathname of the user’s home directory, where a .mdbrc file may reside. If this variable is not present, no .mdbrc processing will occur.

HOME  This variable is used to determine the pathname of the shell used to process shell escapes requested using the ! meta-character. If this variable is not present, /bin/sh is used.

SHELL  This variable is used to determine the maximum length of the command history list. If this variable is not present, the default length is 128.

FILES  User mdb initialization file. The .mdbrc file, if present, is processed after the debug target has been initialized, but before module auto-loading is performed or any commands have been read from standard input.

$HOME/.mdbrc
/dev/kmem  Kernel virtual memory image device. This device special file is used as the core file when examining the live operating system.

/dev/ksyms  Kernel symbol table device. This device special file is used as the object file when examining the live operating system.

/proc/pid/*  Process information files that are read when examining and controlling user processes.

/usr/lib/adb
/usr/platform/platform-name/lib/adb
Default directories for macro files that are read with the `$<` and `$<<` dcmds.

`platform-name` is the name of the platform, derived either from information in a core file or crash dump, or from the current machine as if by `uname -i` (see `uname(1)`).

```
/usr/lib/mdb
/usr/platform/platform-name/lib/mdb
```

Default directories for debugger modules that are loaded using the `::load` dcmd.

`platform-name` is the name of the platform, derived either from information in a core file or crash dump, or from the current machine as if by `uname -i` (see `uname(1)`).

```
SEE ALSO
```

adb(1), cmdtool(1), gcore(1), proc(1), pgrep(1), ps(1), stty(1), truss(1),
uname(1), coreadm(1M), dumpadm(1M), savecore(1M), exec(2), fork(2),
_lwp_self(2), pipe(2), vfork(2), dlopen(3DL), elf(3ELF), libkvm(3LIB),
libthread_db(3LIB), libthread(3LIB), signal(3C), signal(3HEAD),
thr_self(3THR), threads(3THR), core(4), proc(4), attributes(5),
largefile(5), ksym(7D), mem(7D)

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWmdb (32-bit)</td>
</tr>
<tr>
<td></td>
<td>SUNWmdbx (64-bit)</td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Evolving</td>
</tr>
</tbody>
</table>

The debugger and its dmods execute in the same address space, and thus it is quite possible that a buggy dmod can cause `mdb` to dump core or otherwise misbehave. The `mdb` resume capability, described above under Signal Handling, provides a limited recovery mechanism for these situations. However, it is not possible for `mdb` to know definitively whether the dmod in question has corrupted only its own state, or the debugger's global state. Therefore a resume operation cannot be guaranteed to be safe, or to prevent a subsequent crash of the debugger. The safest course of action following a resume is to save any important debug information, and then quit and restart the debugger.

The use of the debugger to modify (that is, write to) the address space of live running operating system is extremely dangerous, and may result in a system panic in the event the user damages a kernel data structure.

`mdb` does not provide support for examining process core files that were generated by a release of Solaris preceding Solaris 2.6. If a core file from one operating system release is examined on a different operating system release, the run-time link-editor debugging interface (`librtld_db`) may not be able to initialize. In this case, symbol

```
Linker and Libraries Guide
Solaris Modular Debugger Guide
```

Use of the Error Recovery Mechanism

Use of the Debugger to Modify the Live Operating System

Limitations on Examining Process Core Files

900 man pages section 1: User Commands • Last Revised 12 Dec 2001
information for shared libraries will not be available. Furthermore, since shared mappings are not present in user core files, the text section and read-only data of shared libraries may not match the data that was present in the process at the time it dumped core. Core files from Solaris Intel systems may not be examined on Solaris SPARC systems, and vice-versa.

Crash dumps from Solaris 7 and earlier releases may only be examined with the aid of the libkvm from the corresponding operating system release. If a crash dump from one operating system release is examined using the dmods from a different operating system release, changes in the kernel implementation may prevent some dcmds or walkers from working properly. `mdb` will issue a warning message if it detects this condition. Crash dumps from Solaris Intel systems may not be examined on Solaris SPARC systems, and vice-versa.

`mdb` provides support for debugging both 32-bit and 64-bit programs. Once it has examined the target and determined its data model, `mdb` automatically re-executes the `mdb` binary that has the same data model as the target, if necessary. This approach simplifies the task of writing debugger modules, because the modules that are loaded will use the same data model as the primary target. Only the 64-bit debugger may be used to debug 64-bit target programs. The 64-bit debugger can only be used on a system that is running the 64-bit operating environment.

The debugger may also need to re-execute itself when debugging a 32-bit process that execs a 64-bit process, or vice-versa. The handling of this situation is discussed in more detail under Interaction with Exec, below.

When a controlled process performs a successful `exec(2)`, the behavior of the debugger is controlled by the `::set -o follow_exec_mode` option, as described above. If the debugger and victim process have the same data model, then the "stop" and "follow" modes determine whether `mdb` automatically continues the target or returns to the debugger prompt following the exec. If the debugger and victim process have a different data model, then the "follow" behavior causes `mdb` to automatically re-exec the `mdb` binary with the appropriate data model and to re-attach to the process, still stopped on return from the exec. Not all debugger state is preserved across this re-exec.

If a 32-bit victim process execs a 64-bit program, then "stop" returns to the command prompt, but the debugger is no longer able to examine the process because it is now using the 64-bit data model. To resume debugging, execute the `::release -a dcmd`, `quit mdb`, and then execute `mdb -p pid` to re-attach the 64-bit debugger to the process.

If a 64-bit victim process execs a 32-bit program, then "stop" will return to the command prompt, but the debugger will only provide limited capabilities for examining the new process. All built-in dcmds will work as advertised, but loadable dcmds will not since they do not perform data model conversion of structures. The user should release and re-attach the debugger to the process as described above in order to restore full debugging capabilities.
If the debugger is attached to a process that is stopped by job control (that is, it stopped in response to SIGTSTP, SIGTTIN, or SIGTTOU), the process may not be able to be set running again when it is continued by a continue dcmd. If the victim process is a member of the same session (that is, it shares the same controlling terminal as mdb), mdb attempts to bring the associated process group to the foreground and to continue the process with SIGCONT to resume it from job control stop. When mdb is detached from such a process, it restores the process group to the background before exiting. If the victim process is not a member of the same session, mdb cannot safely bring the process group to the foreground, so it continues the process with respect to the debugger, but the process remains stopped by job control. mdb prints a warning in this case, and the user must issue an "fg" command from the appropriate shell in order to resume the process.

When mdb attaches to a running process, the process is stopped and remains stopped until one of the continue dcmds is applied, or the debugger quits. If the -onostop option is enabled prior to attaching the debugger to a process with -p, or prior to issuing an ::attach or :A command, mdb attaches to the process but does not stop it. While the process is still running, it may be inspected as usual (albeit with inconsistent results) and breakpoints or other tracing flags may be enabled. If the :c or ::cont dcmds are executed while the process is running, the debugger waits for the process to stop. If no traced software events occur, the user can send an interrupt (^C) after :c or ::cont to force the process to stop and return control to the debugger.

mdb releases the current running process (if any) when the :R, ::release, :r, ::run, :q, or ::quit dcmds are executed, or when the debugger terminates as the result of an EOF or signal. If the process was originally created by the debugger using :r or ::run, it will be forcibly terminated as if by SIGKILL when it is released. If the process was already running prior to attaching mdb to it, it will be set running again when it is released. A process may be released and left stopped and abandoned using the ::release -a option.

Symbolic Debugging Information

The ::list, ::offsetof, ::print, and ::sizeof dcmds require that one or more load objects contain compressed symbolic debugging information suitable for use with mdb. This information is currently only available for certain Solaris kernel modules.

Developer Information

The Solaris Modular Debugger Guide provides a more detailed description of mdb features, as well as information for debugger module developers.

The header file <sys/mdb_modapi.h> contains prototypes for the functions in the MDB Module API, and the SUNWmdbdm package provides source code for an example module in the directory /usr/demo/mdb.
NAME
mesg – permit or deny messages

SYNOPSIS
mesg [-n | -y | n | y]

DESCRIPTION
The mesg utility will control whether other users are allowed to send messages via write(1), talk(1), or other utilities to a terminal device. The terminal device affected is determined by searching for the first terminal in the sequence of devices associated with standard input, standard output, and standard error, respectively. With no arguments, mesg reports the current state without changing it. Processes with appropriate privileges may be able to send messages to the terminal independent of the current state.

OPTIONS
The following options are supported:

- n | n  Deny permission to other users to send messages to the terminal. See write(1).
- y | y  Grant permission to other users to send messages to the terminal.

ENVIRONMENT VARIABLES
See environ(5) for descriptions of the following environment variables that affect the execution of mesg: LC_TYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS
The following exit values are returned:

0  if messages are receivable.
1  if messages are not receivable.
2  on error.

FILES
/dev/tty*  terminal devices
/dev/pts/*  terminal devices

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO
talk(1), write(1), attributes(5), environ(5)
The `message` command puts `string` out on the FMLI message line. If there is no string, the `stdin` input to `message` will be used. The output of `message` has a duration (length of time it remains on the message line). The default duration is "transient": it or one of two other durations can be requested with the mutually-exclusive options below.

Messages displayed with `message -p` will replace (change the value of) any message currently displayed or stored via use of the `permanentmsg` descriptor. Likewise, `message -f` will replace any message currently displayed or stored via use of the `framemsg` descriptor. If more than one message in a frame definition file is specified with the `-p` option, the last one specified will be the permanent duration message.

The `string` argument should always be the last argument.

### OPTIONS

- `-t` Explicitly defines a message to have transient duration. Transient messages remain on the message line only until the user presses another key or a CHECKWORLD occurs. The descriptors `itemmsg`, `fieldmsg`, `invalidmsg`, `choicemsg`, the default-if-not-defined value of `oninterrupt`, and FMLI generated error messages (that is, from syntax errors) also output transient duration messages. Transient messages take precedence over both frame messages and permanent messages.

- `-f` Defines a message to have "frame" duration. Frame messages remain on the message line as long as the frame in which they are defined is current. The descriptor `framemsg` also outputs a frame duration message. Frame messages take precedence over permanent messages.

- `-p` Defines a message to have "permanent" duration. Permanent messages remain on the message line for the length of the FMLI session, unless explicitly replaced by another permanent message or temporarily superseded by a transient message or frame message. A permanent message is not affected by navigating away from, or by closing, the frame which generated the permanent message. The descriptor `permanentmsg` also outputs a permanent duration message.

- `-b[num]` Rings the terminal bell `num` times, where `num` is an integer from 1 to 10. The default value is 1. If the terminal has no bell, the screen will flash `num` times instead, if possible.

- `-o` Forces `message` to duplicate its message to `stdout`.

- `-w` Turns on the working indicator.
EXAMPLES

**EXAMPLE 1** A sample output of `message` on the message line:

When a value entered in a field is invalid, ring the bell 3 times and then display
*Invalid Entry: Try again!* on the message line:

```
invalidmsg=`message -b 3 "Invalid Entry: Try again!"`
```

Display a message that tells the user what is being done:

```
done=`message EDITOR has been set in your environment` close
```

Display a message on the message line and stdout for each field in a form (a
pseudo-"field duration" message).

```
fieldmsg=`message -o -f "Enter a filename."`
```

Display a blank transient message (effect is to "remove" a permanent or frame
duration message).

```
done=`message " "` nop
```

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO

sleep(1), attributes(5)

NOTES

If `message` is coded more than once on a single line, it may appear that only the
right-most instance is interpreted and displayed. Use `sleep(1)` between uses of
`message` in this case, to display multiple messages.

`message -f` should not be used in a stand-alone backquoted expression or with the
`init` descriptor because the frame is not yet current when these are evaluated.

In cases where `message -f "string"` is part of a stand-alone backquoted
expression, the context for evaluation of the expression is the previously current
frame. The previously current frame can be the frame that issued the `open` command
for the frame containing the backquoted expression, or it can be a frame given as an
argument when `fmli` was invoked. That is, the previously current frame is the one
whose frame message will be modified.

Permanent duration messages are displayed when the user navigates to the command
line.
mixinctl(1)

NAME
mixinctl – audio mixer control command line application

SYNOPSIS
/usr/sbin/mixinctl [-a | -d dev] [-iv] [-e | -o]

DESCRIPTION
Some audio devices support the audio mixer functionality. See mixer(7I) for a complete description of the audio mixer. The mixinctl command is used to control the mode of the audio mixer and to get information about the audio mixer and the audio device. See audio(7I) for details.

OPTIONS
The following options are supported. If none are specified, option -i is assumed:

- a
The command applies to all audio devices.

- d dev
The dev argument specifies an alternate audio control device for the command to use.

- e
Enables the audio mixer function if the audio device supports it. If supported, the audio mixer may be enabled at any time. The command silently ignores the enable option if the audio mixer is already enabled.

- i
Prints the audio device type information for the device and indicates whether the audio device uses the audio mixer. If the device does use the audio mixer, this option displays the audio mixer's mode.

- o
Turns off the audio mixer function if the audio device supports it. If supported, the audio mixer may be turned off if only one process has the device opened with the O_RDWR flag, or, if two different processes have the device opened, one with the O_RDONLY flag and the other with the O_WRONLY flag. (See open(2).) The command silently ignores the disable option if the audio mixer function is already disabled.

- v
Verbose mode. Prints the audio_info_t structure for the device, along with the device type information. This option implies the -i option.

ENVIRONMENT VARIABLES

AUDIODEV
If the -d and -a options are not specified, the AUDIODEV environment variable is consulted. If set, AUDIODEV will contain the full path name of the user's default audio device. The default audio device will be converted into a control device, and then used. If the AUDIODEV variable is not set, /dev/audioctl is used.

FILES
/dev/audioctl
/dev/sound/{0...n}ctl

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

906 man pages section 1: User Commands • Last Revised 12 Mar 2001
### ATTRIBUTE TYPE | ATTRIBUTE VALUE
---|---
Architecture | SPARC, IA
Availability | SUNWauda
Stability Level | Evolving

**SEE ALSO**

audioconvert(1), audioplay(1), audiorecord(1), open(2), attributes(5),
usb_ac(7D), audio(7I), audio_support(7I), mixer(7I)
The `mkdir` command creates the named directories in mode 777 (possibly altered by the file mode creation mask `umask(1)`).

Standard entries in a directory (for instance, the files “.,” for the directory itself, and “..,” for its parent) are made automatically. `mkdir` cannot create these entries by name. Creation of a directory requires write permission in the parent directory.

The owner-ID and group-ID of the new directories are set to the process’s effective user-ID and group-ID, respectively. `mkdir` calls the `mkdir(2)` system call.

**setgid and mkdir**

To change the `setgid` bit on a newly created directory, you must use `chmod g+s` or `chmod g-s` after executing `mkdir`.

The `setgid` bit setting is inherited from the parent directory.

**OPTIONS**

The following options are supported:

- `-m mode` This option allows users to specify the mode to be used for new directories. Choices for modes can be found in `chmod(1)`.

- `-p` With this option, `mkdir` creates `dir` by creating all the non-existing parent directories first. The mode given to intermediate directories will be the difference between 777 and the bits set in the file mode creation mask. The difference, however, must be at least 300 (write and execute permission for the user).

**OPERANDS**

The following operand is supported:

- `dir` A path name of a directory to be created.

**EXAMPLES**

**EXAMPLE 1** Using `mkdir`

The following example:

```
example$ mkdir -p ltr/jd/jan
```

creates the subdirectory structure `ltr/jd/jan`.

**ENVIRONMENT VARIABLES**

See `environ(5)` for descriptions of the following environment variables that affect the execution of `mkdir`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

**EXIT STATUS**

The following exit values are returned:

- `0` All the specified directories were created successfully or the `-p` option was specified and all the specified directories now exist.
mkdir(1)

An error occurred.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

SEE ALSO

rm(1), sh(1), umask(1), intro(2), mkdir(2), attributes(5), environ(5), largefile(5)
The `mkmsgs` utility is used to create a file of text strings that can be accessed using the text retrieval tools (see `gettext(1)`, `srchtxt(1)`, `exstr(1)`, and `gettext(3C)`). It will take as input a file of text strings for a particular geographic locale (see `setlocale(3C)`) and create a file of text strings in a format that can be retrieved by both `gettext(1)` and `gettext(3C)`. By using the `-i` option, you can install the created file under the `/usr/lib/locale/locale/LC_MESSAGES` directory (`locale` corresponds to the language in which the text strings are written).

`inputstrings` is the name of the file that contains the original text strings. `msgfile` is the name of the output file where `mkmsgs` writes the strings in a format that is readable by `gettext(1)` and `gettext(3C)`. The name of `msgfile` can be up to 14 characters in length, but may not contain either `\0` (null) or the ASCII code for `/` (slash) or `:` (colon).

The input file contains a set of text strings for the particular geographic locale. Text strings are separated by a newline character. Nongraphic characters must be represented as alphabetic escape sequences. Messages are transformed and copied sequentially from `inputstrings` to `msgfile`. To generate an empty message in `msgfile`, leave an empty line at the correct place in `inputstrings`.

Strings can be changed simply by editing the file `inputstrings`. New strings must be added only at the end of the file; then a new `msgfile` file must be created and installed in the correct place. If this procedure is not followed, the retrieval function will retrieve the wrong string and software compatibility will be broken.

The following options are supported:

- `-o` Overwrite `msgfile`, if it exists.
- `-i locale` Install `msgfile` in the `/usr/lib/locale/locale/LC_MESSAGES` directory. Only someone who is super user or a member of group `bin` can create or overwrite files in this directory. Directories under `/usr/lib/locale` will be created if they do not exist.

### EXAMPLES

**EXAMPLE 1** Using the `mkmsgs` command.

The following example shows an input message source file `C.str`:

```plaintext
File %s:	 cannot be opened
%s: Bad directory
...
write error
...
```
EXAMPLE 1 Using the mkmsgs command.  

EXAMPLE 2 Using Input Strings From C.str to Create Text Strings in a File

The following command uses the input strings from C.str to create text strings in the appropriate format in the file UX in the current directory:

example% mkmsgs C.str UX

EXAMPLE 3 Using Input Strings From FR.str to Create Text Strings in a File

The following command uses the input strings from FR.str to create text strings in the appropriate format in the file UX in the directory /usr/lib/locale/fr/LC_MESSAGES:

example% mkmsgs -i fr FR.str UX

These text strings would be accessed if you had set the environment variable LC_MESSAGES=fr and then invoked one of the text retrieval tools listed at the beginning of the DESCRIPTION section.

FILES
/usr/lib/locale/locale/LC_MESSAGES/*
message files created by mkmsgs

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWloc</td>
</tr>
</tbody>
</table>

SEE ALSO
exstr(1), gettext(1), srchtxt(1), gettext(3C), setlocale(3C), attributes(5)
The `mkstr` utility creates files of error messages. You can use `mkstr` to make programs with large numbers of error diagnostics much smaller, and to reduce system overhead in running the program — as the error messages do not have to be constantly swapped in and out.

`mkstr` processes each of the specified filenames, placing a massaged version of the input file in a file with a name consisting of the specified prefix and the original source file name. A typical example of using `mkstr` would be:

```
mkstr pistrings processed *.c
```

This command would cause all the error messages from the C source files in the current directory to be placed in the file `pistrings` and processed copies of the source for these files to be placed in files whose names are prefixed with `processed`.

To process the error messages in the source to the message file, `mkstr` keys on the string `'error("'` in the input stream. Each time it occurs, the C string starting at the `'"' is placed in the message file followed by a null character and a NEWLINE character; the null character terminates the message so it can be easily used when retrieved, the NEWLINE character makes it possible to sensibly `cat` the error message file to see its contents. The massaged copy of the input file then contains a `lseek` pointer into the file which can be used to retrieve the message, that is:

```
char efilname[ ] = "/usr/lib/pi_strings";
int efil = -1;

error(a1, a2, a3, a4)
{
    char
    buf[256];
    if (efil < 0) {
        efil = open(efilname, 0);
        if (efil < 0) {
            oops:
                perror (efilname);
                exit (1);
        }
    }
    if (lseek(efil, (long) a1, 0) | | read(efil, buf, 256) <= 0)
        goto oops;
    printf(buf, a2, a3, a4);
}
```

Options

```
- Place error messages at the end of the specified message file for recompiling part of a large mkstr ed program.
```

Attributes

See attributes(5) for descriptions of the following attributes:
### CKM Attribute Reference

#### SUNWscpu

<table>
<thead>
<tr>
<th>Attribute Type</th>
<th>Attribute Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

**SEE ALSO**

xstr(1), attributes(5)
Description

The `more` utility is a filter that displays the contents of a text file on the terminal, one screenful at a time. It normally pauses after each screenful. `more` then prints `More--` and `file` at the bottom of the screen. If `more` is reading from a file rather than a pipe, the percentage of characters displayed so far is also shown.

The `more` utility scrolls up to display one more line in response to a RETURN character. `more` displays another screenful in response to a SPACE character. Other commands are listed below.

The `page` utility clears the screen before displaying the next screenful of text. `page` only provides a one-line overlap between screens.

The `more` utility sets the terminal to NOECHO mode, so that the output can be continuous. Commands that you type do not normally show up on your terminal, except for the `/` and `!` commands.

The `/usr/bin/more` utility exits after displaying the last specified file. `/usr/xpg4/bin/more` prompts for a command at the last line of the last specified file.

If the standard output is not a terminal, `more` acts just like `cat(1)`, except that a header is printed before each file in a series.

Options

The following options are supported for both `/usr/bin/more` and `/usr/xpg4/bin/more`:

`-c` Clears before displaying. Redraws the screen instead of scrolling for faster displays. This option is ignored if the terminal does not have the ability to clear to the end of a line.

`-d` Displays error messages rather than ringing the terminal bell if an unrecognized command is used. This is helpful for inexperienced users.

`-s` Squeeze. Replaces multiple blank lines with a single blank line. This is helpful when viewing `nroff(1)` output on the screen.

`/usr/bin/more` The following options are supported for `/usr/bin/more` only:
Does not fold long lines. This is useful when lines contain nonprinting characters or escape sequences, such as those generated when `nroff(1)` output is piped through `ul(1)`.

Does not treat FORMFEED characters (Control-l) as page breaks. If `-l` is not used, `more` pauses to accept commands after any line containing a `^L` character (Control-l). Also, if a file begins with a FORMFEED, the screen is cleared before the file is printed.

Normally, `more` ignores control characters that it does not interpret in some way. The `-r` option causes these to be displayed as `^C` where `C` stands for any such control character.

Suppresses generation of underlining escape sequences. Normally, `more` handles underlining, such as that produced by `nroff(1)`, in a manner appropriate to the terminal. If the terminal can perform underlining or has a stand-out mode, `more` supplies appropriate escape sequences as called for in the text file.

Normally, `more` exits when it comes to the end of its input. With `-w`, however, `more` prompts and waits for any key to be struck before exiting.

Displays the indicated number of `lines` in each screenful, rather than the default (the number of lines in the terminal screen less two).

Start up at `linenumber`.

Start up two lines above the line containing the regular expression `pattern`. Note: Unlike editors, this construct should not end with a `'/.'`. If it does, then the trailing slash is taken as a character in the search pattern.

The following options are supported for `/usr/xpg4/bin/more` only:

Exits immediately after writing the last line of the last file in the argument list.

Performs pattern matching in searches without regard to case.

Specifies the number of lines per screenful. The `number` argument is a positive decimal integer. The `-n` option overrides any values obtained from the environment.

For each file examined, initially executes the `more` command in the `command` argument. If the command is a positioning command, such as a line number or a regular expression search, set the current position to represent the final results of the command, without writing any intermediate lines of the file. For example, the two commands:
more(1)

more -p 1000j file
more -p 1000G file
are equivalent and start the display with the current position at line 1000, bypassing the lines that \( j \) would write and scroll off the screen if it had been issued during the file examination. If the positioning command is unsuccessful, the first line in the file will be the current position.

\(-t\) tagstring

Writes the screenful of the file containing the tag named by the \( tagstring \) argument. See the ctags(1) utility.

\(-u\)

Treats a backspace character as a printable control character, displayed as a \(^H\) (Control-h), suppressing backspacing and the special handling that produces underlined or standout-mode text on some terminal types. Also, does not ignore a carriage-return character at the end of a line.

If both the \(-t\) \( tagstring \) and \(-p\) \( command \) (or the obsolescent \(+command\) ) options are given, the \(-t\) \( tagstring \) is processed first.

Environment

more uses the terminal’s \( \text{terminfo}(4) \) entry to determine its display characteristics.

more looks in the environment variable \( \text{MORE} \) for any preset options. For instance, to page through files using the \(-c\) mode by default, set the value of this variable to \(-c\). (Normally, the command sequence to set up this environment variable is placed in the \( .login \) or \( .profile \) file).

Commands

The commands take effect immediately. It is not necessary to type a carriage return unless the command requires a \( file, command, tagstring, \) or \( pattern \) argument. Up to the time when the command character itself is given, the user may type the line kill character to cancel the numerical argument being formed. In addition, the user may type the erase character to redisplay the ‘\(-\--More--\( xx\%\)\)’ or \( file \) message.

In the following commands, \( i \) is a numerical argument (1 by default).

\( i\)SPACE

Display another screenful, or \( i \) more lines if \( i \) is specified.

\( i\)RETURN

Display another line, or \( i \) more lines, if specified.

\( i\)\( b\)

\( i\)^B

(Control-b) Skip back \( i \) screenfuls and then print a screenful.

\( i\)d

\( i\)^D

(Control-d) Scroll forward one half screenful or \( i \) more lines. If \( i \) is specified, the count becomes the default for subsequent \( d \) and \( u \) commands.

\( i\)f

Skip \( i \) screens full and then print a screenful.

\( h\)

Help. Give a description of all the more commands.

\(^L\)

(Control-l) Refresh.
in
Search for the $i$th occurrence of the last pattern entered.

q
Exit from more.

Q
Exit from more (same as q or Q).

is
Skip $i$ lines and then print a screenful.

v
Drop into the vi editor at the current line of the current file.

iz
Same as SPACE, except that $i$, if present, becomes the new default number of lines per screenful.

=
Display the current line number.

i/pattern
Search forward for the $i$th occurrence of the regular expression pattern. Display the screenful starting two lines before the line that contains the $i$th match for the regular expression pattern, or the end of a pipe, whichever comes first. If more is displaying a file and there is no match, its position in the file remains unchanged. Regular expressions can be edited using erase and kill characters. Erasing back past the first column cancels the search command.

:command
Invoke a shell to execute command. The characters % and !, when used within command are replaced with the current filename and the previous shell command, respectively. If there is no current filename, % is not expanded. Prepend a backslash to these characters to escape expansion.

:f
Display the current filename and line number.

i:n
Skip to the $i$th next filename given in the command line, or to the last filename in the list if $i$ is out of range.

i:p
Skip to the $i$th previous filename given in the command line, or to the first filename if $i$ is out of range. If given while more is positioned within a file, go to the beginning of the file. If more is reading from a pipe, more simply rings the terminal bell.

:/usr/bin/more
The following commands are available only in /usr/bin/more:

'
Single quote. Go to the point from which the last search started. If no search has been performed in the current file, go to the beginning of the file.

.
Dot. Repeat the previous command.

^\nHalt a partial display of text. more stops sending output, and displays the usual --More-- prompt. Some output is lost as a result.

:/usr/xpg4/bin/more
The following commands are available only in /usr/xpg4/bin/more:

i^F
(Control-f) Skip $i$ screens full and print a screenful. (Same as if.)
```
more(1)

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>^G</td>
<td>(Control-g) Display the current line number (same as =).</td>
</tr>
<tr>
<td>ig</td>
<td>Go to line number i with the default of the first line in the file.</td>
</tr>
<tr>
<td>iG</td>
<td>Go to line number i with the default of the Last line in the file.</td>
</tr>
<tr>
<td>ij</td>
<td>Display another line, or i more lines, if specified. (Same as i\RETURN.)</td>
</tr>
<tr>
<td>i\k</td>
<td>Scroll backwards one or i lines, if specified.</td>
</tr>
<tr>
<td>ml</td>
<td>Mark the current position with the name letter.</td>
</tr>
<tr>
<td>N</td>
<td>Reverse direction of search.</td>
</tr>
<tr>
<td>r</td>
<td>Refresh the screen.</td>
</tr>
<tr>
<td>R</td>
<td>Refresh the screen, discarding any buffered input.</td>
</tr>
<tr>
<td>iu</td>
<td>(Control-u) Scroll backwards one half a screen of i lines, if specified. If i is specified, the count becomes the new default for subsequent d and u commands.</td>
</tr>
<tr>
<td>i^U</td>
<td>(Control-u) Scroll backwards one half a screen of i lines, if specified. If i is specified, the count becomes the new default for subsequent d and u commands.</td>
</tr>
<tr>
<td>ZZ</td>
<td>Exit from more (same as q).</td>
</tr>
<tr>
<td>:e file</td>
<td>Examine (display) a new file. If no file is specified, the current file is redisplayed.</td>
</tr>
<tr>
<td>:t tagstring</td>
<td>Go to the tag named by the tagstring argument and scroll/rewrite the screen with the tagged line in the current position. See the ctags utility.</td>
</tr>
<tr>
<td>' letter</td>
<td>Return to the position that was previously marked with the name letter.</td>
</tr>
<tr>
<td>'</td>
<td>Return to the position from which the last move of more than a screenful was made. Defaults to the beginning of the file.</td>
</tr>
<tr>
<td>i?][/pattern</td>
<td>Search backward in the file for the ith line containing the pattern. The ! specifies to search backward for the ith line that does not contain the pattern.</td>
</tr>
<tr>
<td>i/ \pattern</td>
<td>Search forward in the file for the ith line that does not contain the pattern.</td>
</tr>
<tr>
<td>![command]</td>
<td>Invoke a shell or the specified command.</td>
</tr>
</tbody>
</table>

Large File Behavior

See largefile(5) for the description of the behavior of more and page when encountering files greater than or equal to 2 Gbyte (2^31 bytes).

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of more: LC_COLLATE (/usr/xpg4/bin/more only), LC_CTYPE, LC_MESSAGES, NLSPATH, and TERM.

918 man pages section 1: User Commands • Last Revised 18 Mar 1997
The following environment variables also affect the execution of
/usr/xpg4/bin/more:

- **COLLUMNS**: Overrides the system selected horizontal screen size.
- **EDITOR**: Used by the `v` command to select an editor.
- **LINES**: Overrides the system selected vertical screen size. The `-n` option has precedence over `LINES` in determining the number of lines in a screen.
- **MORE**: A string specifying options as described in the OPTIONS section, above. As in a command line, the options must be separated by blank characters and each option specification must start with a `−`. Any command line options are processed after those specified in `MORE` as though the command line were: `more $MORE options operands`

**EXIT STATUS**
The following exit values are returned:

- **0**: Successful completion.
- **>0**: An error occurred.

**FILES**
```
/usr/lib/more.help   help file for /usr/bin/more and /usr/bin/page
```

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

```
<table>
<thead>
<tr>
<th>/usr/bin/more /usr/bin/page</th>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWCsu</td>
<td></td>
</tr>
<tr>
<td>CSI</td>
<td>Not enabled</td>
<td></td>
</tr>
</tbody>
</table>

```

```
<table>
<thead>
<tr>
<th>/usr/xpg4/bin/more</th>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWxCu4</td>
<td></td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
<td></td>
</tr>
</tbody>
</table>

```

**SEE ALSO**
cat(1), csh(1), ctags(1), man(1), nroff(1), script(1), sh(1), ul(1), environ(4), terminfo(4), attributes(5), environ(5), largefile(5)

```
regcomp(3C)
regex(5), XPG4(5)
```

Skipping backwards is too slow on large files.

This utility will not behave correctly if the terminal is not set up properly.
The mp program, when called without the -D or -P option, reads each filename in
sequence and generates a prettified version of the contents in PostScript™ format, sent
to standard output. If no filename argument is provided, mp reads the standard input.
If the standard input is a terminal, input is terminated by an EOF signal, usually
Control-d.

The -D and -P options require the target printer name as an argument and produce
the Printer Description Language (PDL) of the target printer. The -D option causes the
PDL to output to stdout and the -P option causes the PDL to be directly spooled to the
printer. In the absence of these options, mp will produce default PostScript™ output.

The mp program accepts international text files of various Solaris locales and produces
output which is proper for the specified locale. The output will also contain proper
text layout. For instance, the output will contain bidirectional text rendering, and also
shaping, since the complex text layout (CTL) is supported in mp.

Mail items, news articles, ordinary ASCII files, complete mail folders, and digests are
all acceptable input formats for mp. The output format includes grayscale lozenges, or
the outline of the same dimensions as the lozenges, containing banner information at
the top and bottom of every page.

OPTIONS
The following options are supported:

- **-a**  Formats the file as a news article. The top banner
  contains the text: "Article from newsgroup", where
  newsgroup is the first news group found on the
  "Newsgroups:" line.

- **-A4**  Uses A4 paper size (8.26 x 11.69 inches).

- **-c chars**  The maximum number of characters to extract from the
  gecos field of the user’s /etc/passwd entry. The
default is 18.

- **-C**  Instead of using "\nFrom" to denote the start of new
  mail messages, mp will look for (and use) the value of
  the Content-Length: mail header. If the
  Content-Length doesn’t take you to the next "\nFrom",
  then it is wrong, and mp falls back to looking for the
  next "\nFrom" in the mail folder.

- **-d**  Formats the file as a digest.
mp(1)

-D target_printer_name

Produces the PDL for the target printer. Requires X Print Server connection.

-e

Assumes the ELM mail frontend intermediate file format. Used when printing messages from within ELM (using the "p" command), especially for printing tagged messages. This option must be specified in your ELM option setup.

-ff

Formats the file for use with a Filofax personal organizer.

-fp

Formats the file for use with a Franklin Planner personal organizer.

-F

Instead of printing who the mail article is for, the top header will contain who the mail article is from. A useful option for people with their own personal printer.

-1

Formats output in landscape mode. Two pages of text will be printed per sheet of paper.

-ll

Formats output in landscape mode. One page of text will be printed per sheet of paper. This is useful for printing files with longer than normal lines.

-L localename

Provides the locale of the file to be printed. If this command line option is not present, then mp looks for the MP_LANG environment variable. If that is not present, then the LANG environment variable is used. If none of these options are present, then mp tries to determine the locale it is running in, and if it cannot, then it assumes it is running in the C locale.

-m

Formats the file as a mail folder, printing multiple messages.

-M

Forces mp to use the mp.conf file for printing output even if a prolog.ps file exists for that locale. Useful when printing to non-native PostScript™ printers.

-n

Turns off the gray bars and associated information from header and footer. Used to get output similar to output of 'lp filename'.

-o

Formats the file as an ordinary ASCII file.

-p prologue

Employs the file prologue as the PostScript/Xprt prologue file, overriding any previously defined file names. This file specifies the format of the print output. For PostScript output, the prologue file will have a .ps extension. For Xprt clients (when the -D option is
specified), this file will have an .xpr extension. These files are defined in the SUPPLIED PROLOGUE FILES section below.

-\(P\) \texttt{target\_spool\_printer} Spools the PDL to the target printer. No output is sent to stdout. Requires X Print Server connection.

-\(PS\) If the mail or digest message just has PostScript as the text of the message, this is normally just passed straight through. Specifying this option causes PostScript to be printed as text.

-\(s\) \texttt{subject} Uses \texttt{subject} as the new subject for the printout. If you are printing ordinary ASCII files that have been specified on the command line, the subject will default to the name of each of these files.

-\(tm\) Formats the file for use with the Time Manager personal organizer.

-\(ts\) Formats the file for use with the Time/System International personal organizer.

-\(US\) Uses US paper size (8.5 x 11 inches). This is the default paper size.

-\(u\) \texttt{config\_file\_path} Specifies an alternate configuration file to the default file

\[
\texttt{/usr/lib/lp/locale/locale\_name/mp/mp.conf.}\n\]

The absolute file path name must be used.

-\(v\) Prints the version number of this release of \texttt{mp}.

-\(w\) \texttt{words} The maximum number of words to extract from the gecos field of the user’s /etc/passwd entry. The default is 3.

-\(z\) \texttt{point\_size} Prints the output text in the point size specified by \texttt{point\_size}. The internal default is 12 points for portrait printing and 9 points for landscape printing.

-\(?\) Prints the usage line for \texttt{mp} (notice that the \texttt{?} character must be escaped if using \texttt{csh}(1)).

**OPERANDS**

The following operand is supported:

\texttt{filename} The name of the file to be read.

**ENVIRONMENT VARIABLES**

\texttt{XPDISPLAY} If either the -\(D\) or -\(P\) option is specified in the command line and there is no \texttt{XPDISPLAY} variable set in the environment, the print server startup script will start an Xprt server at port 2100 in the machine in which the client is running. The script will terminate the print server once the job is over. If \texttt{XPDISPLAY} is set, the \texttt{mp}
client will try to contact the print server running at `XPDISPLAY`. In this case, no attempt will be made to start the server if it is not running.

**MP_PROLOGUE**

Used to determine the directory where the page formatting files (.xpr or .ps) are kept. These files determine page decorations, number of logical pages per physical page, landscape or portrait format, and so forth. In the absence of MP_PROLOGUE, the default location of the directory is `/usr/lib/lp/locale/C/mp`.

**MP_LANG**

If neither of the `-D` or `-P` options is specified, a prologue file is prepended to the output to be printed. The prologue file is called `/usr/openwin/lib/locale/localename/print/prolog.ps` or `/usr/lib/lp/locale/localename/mp/prolog.ps`, where `localename` is the value of the MP_LANG or LANG environment variable, if present. If both variables are present, the file `/usr/openwin/lib/locale/localename/print/prolog.ps` is given preference due to backward compatibility reasons. If either of these files are not present, and the `-D` option is not specified, a configuration file of the locale called `/usr/lib/lp/locale/localename/mp/mp.conf` is used as the source of the configuration information that substitutes the prologue information for printing. The presence of `prolog.ps` disables `mp.conf` for backward compatibility.

### EXIT STATUS

The following exit values are returned:

- **0** Successful completion.
- **1** An error occurred.

### SUPPLIED PROLOGUE FILES

The following prologue files are provided. Files with .ps extensions are for the PostScript output. Files with .xpr extensions are for the Print Server client. .xpr files are created for 300dpi printers and will scale to other resolution values.

- `mp.common.ps` Common prologue file for all other .ps files in this directory.
- `mp.pro.ps` Used by default.
- `mp.pro.xpr` Used if the `-ff` option is in effect.
- `mp.pro.fp.ps` Used if the `-fp` option is in effect.
- `mp.pro.tm.ps` Used if the `-tm` option is in effect.
mp(1)

<table>
<thead>
<tr>
<th>mp.pro.ts.ps</th>
<th>Used if the -ts option is in effect.</th>
</tr>
</thead>
<tbody>
<tr>
<td>mp.pro.ts.xpr</td>
<td></td>
</tr>
<tr>
<td>mp.pro.alt.ps</td>
<td>An alternative modification of the default prologue file which outputs the page number in the right corner of the bottom banner.</td>
</tr>
<tr>
<td>mp.pro.alt.xpr</td>
<td></td>
</tr>
<tr>
<td>mp.pro.l.ps</td>
<td>Prologue file used for landscape outputs.</td>
</tr>
<tr>
<td>mp.pro.l.xpr</td>
<td></td>
</tr>
<tr>
<td>mp.pro.ll.ps</td>
<td>Prologue file used for landscape outputs, when printing files with longer than normal lines.</td>
</tr>
<tr>
<td>mp.pro.ll.xpr</td>
<td></td>
</tr>
<tr>
<td>mp.pro.altl.ps</td>
<td>Alternate prologue file used for landscape outputs.</td>
</tr>
<tr>
<td>mp.pro.altl.xpr</td>
<td></td>
</tr>
</tbody>
</table>

FILES

.cshrc
  Initialization file for csh(1).

.mailrc
  Initialization file for mail(1).

/usr/bin/mp
  Executable.

/usr/lib/lp/locale/C/mp/mp.conf
  Default configuration file.

/usr/lib/lp/locale/C/mp/mp.common.ps
  Common prologue file for all other .ps files in this directory. Not for .xpr files.

/usr/lib/lp/locale/C/mp/mp.pro.ps
/usr/lib/lp/locale/C/mp/mp.pro.xpr
  Default prologue files for mail printing.

/usr/lib/lp/locale/C/mp/mp.pro.l.ps
/usr/lib/lp/locale/C/mp/mp.pro.l.xpr
  Default prologue files for landscape format.

/usr/lib/lp/locale/C/mp/mp.pro.ll.ps
/usr/lib/lp/locale/C/mp/mp.pro.ll.xpr
  Default prologue files for landscape format with one column per page. Useful when printing files with long lines.

/usr/lib/lp/locale/C/mp/mp.pro.altl.ps
/usr/lib/lp/locale/C/mp/mp.pro.altl.xpr
  Alternate prologue files for landscape format.

/usr/lib/lp/locale/C/mp/mp.pro.alt.ps
/usr/lib/lp/locale/C/mp/mp.pro.alt.xpr
  Alternative “default” prologue files. Insert page numbers in the bottom right corner of each page.
Default prologue files for Filofax format.

Default prologue files for Franklin Planner format.

Default prologue files for Time Manager format.

Default prologue files for Time/System International format.

Default locale-specific prologue file as an alternative to the mp.conf file. See ENVIRONMENT VARIABLES for more detail on the relationship.

The structure and format for mp.conf and .xpr files are documented in the International Language Environments Guide. Refer to this document if you need to use alternate fonts, including Printer Resident Fonts, or if you want to make changes to output format.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWmp</td>
</tr>
</tbody>
</table>

SEE ALSO
csh(1), mail(1), mailtool(1), attributes(5)

International Language Environments Guide
The `mpss.so.1` shared object provides a means by which the preferred stack and/or heap page size can be selectively configured for launched processes and their descendants. To enable `mpss.so.1`, the following string needs to be present in the environment (see `ld.so.1(1)`) along with one or more MPSS (Multiple Page Size Support) environment variables:

```
LD_PRELOAD=$LD_PRELOAD:mpss.so.1
```

Once preloaded, the `mpss.so.1` shared object reads the following environment variables to determine any preferred page size requirements and any processes these may be specific to.

- **MPSSHEAP=size**
  - `MPSSHEAP` and `MPSSSTACK` specify the preferred page sizes for the heap and stack, respectively. The specified page size(s) are applied to all created processes.
  - `size` must be a supported page size (see `pagesize(1)`) or 0, in which case the system will select an appropriate page size (see `mmap>(2)`).
  - `size` can be qualified with K, M, G, or T to specify Kilobytes, Megabytes, Gigabytes, or Terabytes respectively.

- **MPSSSTACK=size**

- **MPSSCFGFILE=configuration file**
  - `MPSSCFGFILE` is a text file which contains one or more `mpss` configuration entries of the form:
  ```
  exec-name:heap-size:stack-size
  ```
  - `exec-name` specifies the name of an application or executable. The corresponding preferred page size(s) are set for newly created processes (see `getexecname>(3C)`) that match the first `exec-name` found in the file.
  - `exec-name` can be a full pathname, a base name or a pattern string. See File Name Generation in `sh>(1)` for a discussion of pattern matching.
  - If `heap-size` and/or `stack-size` are not specified, the corresponding preferred page size(s) will not be set.

- **MPSSERRFILE=pathname**
  - By default, error messages are logged via `syslog>(3C)` using level `LOG_ERR` and facility `LOG_USER`. If
MPSSERRFILE contains a valid pathname (such as /dev/stderr), error messages will be logged there instead.

**EXAMPLE 1** Configuring preferred page sizes using MPSSCFGFILE

The following Bourne shell commands (see `sh(1)`) configure the preferred page sizes to a select set of applications with exec names that begin with `foo`, using the MPSSCFGFILE environment variable. The MPSS configuration file, `mpsscfg`, is assumed to have been previously created via a text editor like `vi(1)`. The `cat(1)` command is only dumping out the contents.

```bash
example$ LD_PRELOAD=$LD_PRELOAD:mpss.so.1
example$ MPSSCFGFILE=mpsscfg
example$ export LD_PRELOAD MPSSCFGFILE
example$ cat $MPSSCFGFILE
foo*:512K:64K
```

Once the application has been started, `pmap` (see `proc(1)`) can be used to view the actual page sizes configured:

```bash
example$ foobar &
example$ pmap -s 'pgrep foobar'
```

If the desired page size is not configured (shown in the `pmap` output), it may be due to errors in the MPSS configuration file or environment variables. Check the error log (by default: `/var/adm/messages`) for errors.

If no errors can be found, resource or alignment constraints may be responsible. See the NOTES section.

**EXAMPLE 2** Configuring preferred page sizes using MPSSHAP and MPSSSTACK

The following Bourne shell commands configure 512K heap and 64K stack preferred page sizes for all applications using the MPSSHAP and MPSSSTACK environment variables.

```bash
example$ LD_PRELOAD=$LD_PRELOAD:mpss.so.1
example$ MPSSHAP=512K
example$ MPSSSTACK=64K
example$ export LD_PRELOAD MPSSHAP MPSSSTACK
```

**EXAMPLE 3** Precedence rules (continuation from Example 2)

The preferred page size configuration in MPSSCFGFILE overrides MPSSHAP and MPSSSTACK. Appending the following commands to those in Example 2 would mean that all applications will be configured with 512K heap and 64K stack preferred page sizes with the exception of those applications, the `ls` command, and all applications beginning with `ora`, in the configuration file.
EXAMPLE 3 Precedence rules (continuation from Example 2)  (Continued)

example$ MPSCFGFILE=mpsscfg2
example$ export MPSCFGFILE
example$ cat $MPSCFGFILE
ls:
ora*:4m:4m

FILES
/usr/lib/ld/map.bssalign  A template link-editor mapfile for aligning bss (see NOTES).

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu (32-bit)</td>
</tr>
<tr>
<td></td>
<td>SUNWesxu (64-bit)</td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Evolving</td>
</tr>
</tbody>
</table>

SEE ALSO
cat(1), ld(1), ld.so.1(1), pagesize(1), ppgsz(1), proc(1), sh(1), vi(1), exec(2), fork(2), memcntl(2), getexecname(3C), getpagesize(3C), syslog(3C), proc(4), attributes(5)

NOTES
The heap and stack preferred page sizes are inherited. A child process has the same preferred page sizes as its parent. On exec(2), the preferred page sizes are set back to the default system page size unless a preferred page size has been configured via the mpss shared object.

ppgsz(1), a proc tool, can also be used to set the preferred stack and/or heap page sizes. It cannot selectively configure the page size for descendents based on name matches.

See also NOTES under ppgsz(1).
## NAME
msgfmt – create a message object from a message file

## SYNOPSIS
```
msgfmt [-D dir | --directory=dir] [-f | --use-fuzzy] [-g]
       [--verbose] filename.po...
```

## DESCRIPTION
The `msgfmt` utility creates message object files from portable object files (filename.po), without changing the portable object files.

The .po file contains messages displayed to users by system commands or by application programs. .po files can be edited. The messages in these files can be rewritten in any language supported by the system.

The `xgettext(1)` command can be used to create .po files from script or programs.

`msgfmt` interprets data as characters according to the current setting of the LC_CTYPE locale category or according to the codeset specified in the .po file.

## OPTIONS
The following options are supported:

- `-D dir`
  - `--directory=dir`
    Adds dir to the list for input files search.

- `-f`
  - `--use-fuzzy`
    Uses fuzzy entries in output. If this option is not specified, fuzzy entries are not included into the output. These options are ignored if Solaris message catalogs are processed.

- `-g`
  Directs the utility to generate the GNU-compatible message catalog file. This option cannot be specified with the -s option.

- `-o output-file`
  - `--output-file=output-file`
    Specifies the output file name as output-file. All domain directives and duplicate msgids in the .po file are ignored.

- `-s`
  Directs the utility to generate the Solaris message catalog file. This option cannot be specified with the -g option.

- `--strict`
  Directs the utility to append the suffix .mo to the generating message object file name if it doesn’t have this suffix. This option is ignored if Solaris message catalogs are processed.

- `-v`
  - `--verbose`
    Verbose. Lists duplicate message identifiers if Solaris message catalog files are processed. Message strings are not redefined.
If GNU-compatible message files are processed, this option detects and diagnoses input file anomalies which might represent translation errors. The msgid and msgstr strings are studied and compared. It is considered abnormal if one string starts or ends with a newline while the other does not. Also, if the string represents a format string used in a printf-like function, both strings should have the same number of % format specifiers, with matching types. If the flag c-format appears in the special comment ‘#’ for this entry, a check is performed.

**USAGE**

The format of portable object files (.po files) is defined as follows. Each .po file contains one or more lines, with each line containing either a comment or a statement. Comments start the line with a pound sign (#) and end with the newline character. All comments (except special comments described later) and empty lines are ignored. The format of a statement is:

```
directive value
```

Each directive starts at the beginning of the line and is separated from value by white space (such as one or more space or tab characters). value consists of one or more quoted strings separated by white space. Use any of the following types of directives for the Solaris message file:

- `domain domainname`
- `msgid message_identifier`
- `msgstr message_string`

For a GNU-compatible message file, use any of the following types of directives:

- `domain domainname`
- `msgid message_identifier`
- `msgid_plural untranslated_string_plural`
- `msgstr message_string`
- `msgstr[n] message_string`

The behavior of the domain directive is affected by the options used. See OPTIONS for the behavior when the -o or --output-file options are specified. If the -o or --output-file options are not specified, the behavior of the domain directive is as follows:

- All msgids from the beginning of each .po file to the first domain directive are put into a default message object file. The default message object file is named messages.mo, if the Solaris message catalog file format is used to generate the message object file or if the --strict option is specified. Otherwise, the default message object file is named messages.
When `msgfmt` encounters a `domain domainname` directive in the `.po` file, all following msgids until the next `domain` directive are put into the message object file, named `domainname.mo`, if the Solaris message catalog file format is used to generate the message object file or if the `--strict` option is specified. Otherwise, the msgids are put into the message object file named `domainname`.

Duplicate msgids are defined in the scope of each domain. That is, a msgid is considered a duplicate only if the identical msgid exists in the same domain.

All duplicate msgids are ignored.

The `msgid` directive specifies the value of a message identifier associated with the directive that follows it. The `msgid_plural` directive specifies the plural form message specified to the plural message handling functions `ngettext()`, `dngettext()`, or `dcngettext()`. The `message_identifier` string identifies a target string to be used at retrieval time. Each statement containing a `msgid` directive must be followed by a statement containing a `msgstr` directive or `msgstr[n]` directives.

The `msgstr` directive specifies the target string associated with the `message_identifier` string declared in the immediately preceding `msgid` directive.

The directive `msgstr[n]` (where `n = 0, 1, 2, ...`) specifies the target string to be used with plural form handling functions `ngettext()`, `dngettext()`, and `dcngettext()`.

Message strings can contain the escape sequences `\n` for newline, `\t` for tab, `\v` for vertical tab, `\b` for backspace, `\r` for carriage return, `\f` for formfeed, `\\` for backslash, `"` for double quote, `\a` for alarm, `\ddd` for octal bit pattern, and `\xDD` for hexadecimal bit pattern.

Comments for a GNU-compatible message file should be in one of the following formats (the `msgfmt` utility will ignore these comments when processing Solaris message files):

```
# translator-comments
#. automatic-comments
#: reference..
#, flag
```

The `#:` comments indicate the location of the msgid string in the source files in `filenameline` format. The `#`, `#:`, and `#:` comments are informative only and are silently ignored by the `msgfmt` utility. The `#:` comments require one or more flags separated by the comma character. The following `flags` can be specified:

**fuzzy**

This flag can be inserted by the translator. It shows that the `msgstr` string might not be a correct translation (anymore). Only the translator can judge if the translation requires further modification or is acceptable as is. Once satisfied with the translation, the translator removes this `fuzzy` flag. If this flag is
specified, the *msgfmt* utility will not generate the entry for the immediately following msgid in the output message catalog.

**c-format**

The *c-format* flag indicates that the msgid string is used as a format string by printf-like functions. In case the *c-format* flag is given for a string, the *msgfmt* utility does some more tests to check the validity of the translation.

In the GNU-compatible message file, the msgid entry with empty string ("") is called the header entry and treated specially. If the message string for the header entry contains nplurals=value, the value indicates the number of plural forms. For example, if nplurals=4, there are four plural forms. If nplurals is defined, the same line should contain plural=expression, separated by a semicolon character. The *expression* is a C language expression to determine which version of msgstr[n] is to be used based on the value of n, the last argument of ngettext(), dgettext(), or dcngettext(). For example,

\[
nplurals=2; plural\ = \ n \ == \ 1 \ ? \ 0 : 1
\]

indicates that there are two plural forms in the language. msgstr[0] is used if n == 1, otherwise msgstr[1] is used. For another example:

\[
nplurals=3; plural\ = \ n \ == \ 1 \ ? \ 0 : n \ == \ 2 \ ? \ 1 : 2
\]

indicates that there are three plural forms in the language. msgstr[0] is used if n == 1, msgstr[1] is used if n == 2, otherwise msgstr[2] is used.

If the header entry contains a charset=codeset string, the *codeset* is used to indicate the codeset to be used to encode the message strings. If the output string's codeset is different from the message string's codeset, codeset conversion from the message string's codeset to the output string's codeset will be performed upon the call of gettext(), dgettext(), dcgettext(), ngettext(), dgettext(), and dcngettext() for the GNU-compatible message catalogs. The output string’s codeset is determined by the current locale’s codeset (the return value of nl_langinfo(CODESET)) by default, and can be changed by the call of bind_textdomain_codeset().

The *msgfmt* utility can generate the message object both in Solaris message catalog file format and in GNU-compatible message catalog file format. If the -s option is specified and the input file is a Solaris .po file, the *msgfmt* utility generates the message object in Solaris message catalog file format. If the -g option is specified and the input file is a GNU .po file, the *msgfmt* utility generates the message object in GNU-compatible message catalog file format. If neither the -s nor -g option is specified, the *msgfmt* utility determines the message catalog file format as follows:

- If the .po file contains a valid GNU header entry (having an empty string for msgid), the *msgfmt* utility uses the GNU-compatible message catalog file format.
- Otherwise, the *msgfmt* utility uses the Solaris message catalog file format.
If the `msgfmt` utility determined that the Solaris message catalog file format is used, as above, but found the `.po` file contains directives that are specific to the GNU-compatible message catalog file format, such as `msgid_plural` and `msgstr[n]`, the `msgfmt` utility handles those directives as invalid specifications.

**EXAMPLE 1** Creating message objects from message files

In this example, `module1.po` and `module2.po` are portable message objects files.

```plaintext
example% cat module1.po
# default domain "messages.mo"
msgid "msg 1"
msgstr "msg 1 translation"
#
# domain "help_domain"
msgid "help 2"
msgstr "help 2 translation"
#
# domain "error_domain"
msgid "error 3"
msgstr "error 3 translation"
example% cat module2.po
# default domain "messages.mo"
msgid "mesg 4"
msgstr "mesg 4 translation"
#
# domain "error_domain"
msgid "error 5"
msgstr "error 5 translation"
#
# domain "window_domain"
msgid "window 6"
msgstr "window 6 translation"
```

The following command will produce the output files `messages.mo`, `help_domain.mo`, and `error_domain.mo` in Solaris message catalog file format:

```plaintext
example% msgfmt module1.po
```

The following command will produce the output files `messages.mo`, `help_domain.mo`, `error_domain.mo`, and `window_domain.mo` in Solaris message catalog file format:

```plaintext
example% msgfmt module1.po module2.po
```

The following command will produce the output file `hello.mo` in Solaris message catalog file format:

```plaintext
example% msgfmt -o hello.mo module1.po module2.po
```

**ENVIRONMENT VARIABLES**

See `environ(5)` for descriptions of the following environmental variables that affect the execution of `msgfmt`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

**ATTRIBUTES**

See `attributes(5)` for descriptions of the following attributes:
Installing message catalogs under the C locale is pointless, since they are ignored for the sake of efficiency.
mt – magnetic tape control

mt [-f tapename] command... [count]

The mt utility sends commands to a magnetic tape drive. If -f tapename is not specified, the environment variable TAPE is used. If TAPE does not exist, mt uses the device /dev/rmt/0n.

The following option is supported:

-f tapename Specifies the raw tape device.

The following operands are supported:

count The number of times that the requested operation is to be performed. By default, mt performs command once. Multiple operations of command may be performed by specifying count.

command Available commands that can be sent to a magnetic tape drive. Only as many characters as are required to uniquely identify a command need be specified.

eof, weof Write count EOF marks at the current position on the tape.

fsf Forward space over count EOF marks. The tape is positioned on the first block of the file.

fsr Forward space count records.

bsf Back space over count EOF marks. The tape is positioned on the beginning-of-tape side of the EOF mark.

bsr Back space count records.

nbsf Back space count files. The tape is positioned on the first block of the file. This is equivalent to count+1 bsf's followed by one fsf.

asf Absolute space to count file number. This is equivalent to a rewind followed by a fsf count.

If count is specified with any of the following commands, the count is ignored and the command is performed only once.

eom Space to the end of recorded media on the tape. This is useful for appending files onto previously written tapes.

rewind Rewind the tape.
Rewind the tape and, if appropriate, take the drive unit off-line by unloading the tape. It cycles through all four tapes.

Print status information about the tape unit.

Rewind the cartridge tape completely, then wind it forward to the end of the reel and back to beginning-of-tape to smooth out tape tension.

Allow the tape drive to remain reserved after closing the device. The drive must then be explicitly released.

Re-establish the default behavior of releasing at close.

Break the reservation of the tape drive held by another host and then reserve the tape drive. This command can be executed only with super-user privileges.

Erase the entire tape. Erasing a tape may take a long time depending on the device and/or tape. Refer to the device specific manual for time details.

All operations were successful.

Command was unrecognized or mt was unable to open the specified tape drive.

An operation failed.

/dev/rmt/*  magnetic tape interface

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO  
tar(1), tcopy(1), ar(3HEAD), environ(4), attributes(5), mtio(7I), st(7D)
Not all devices support all options. Some options are hardware-dependent. Refer to the corresponding device manual page.

mt is architecture sensitive. Heterogeneous operation (that is, SPARC to IA or the reverse) is not supported.
NAME
mv - move files

SYNOPSIS
/usr/bin/mv [-fi] source target_file
/usr/bin/mv [-fi] source... target_dir
/usr/xpg4/bin/mv [-fi] source target_file
/usr/xpg4/bin/mv [-fi] source... target_dir

DESCRIPTION
In the first synopsis form, the mv utility moves the file named by the source operand to
the destination specified by the target_file. source and target_file may not have the same
name. If target_file does not exist, mv creates a file named target_file. If target_file exists,
its contents are overwritten. This first synopsis form is assumed when the final
operand does not name an existing directory.

In the second synopsis form, mv moves each file named by a source operand to a
destination file in the existing directory named by the target_dir operand. The
destination path for each source is the concatenation of the target directory, a single
slash character (/), and the last path name component of the source. This second form
is assumed when the final operand names an existing directory.

If mv determines that the mode of target_file forbids writing, it will print the mode (see
chmod(2)), ask for a response, and read the standard input for one line. If the response
is affirmative, the mv occurs, if permissible; otherwise, the command exits. Notice that
the mode displayed may not fully represent the access permission if target is
associated with an ACL. When the parent directory of source is writable and has the
sticky bit set, one or more of the following conditions must be true:

- the user must own the file
- the user must own the directory
- the file must be writable by the user
- the user must be a privileged user

If source is a file and target_file is a link to another file with links, the other links remain
and target_file becomes a new file.

If source and target_file/target_dir are on different file systems, mv copies the source and
deletes the original. Any hard links to other files are lost. mv will attempt to duplicate
the source file characteristics to the target, that is, the owner and group id, permission
modes, modification and access times, ACLs, and extended attributes, if applicable.
For symbolic links, mv will preserve only the owner and group of the link itself.

If unable to preserve owner and group id, mv will clear S_ISUID and S_ISGID bits in
the target. mv will print a diagnostic message to stderr if unable to clear these bits,
though the exit code will not be affected. mv may be unable to preserve extended
attributes if the target file system does not have extended attribute support.
/usr/xpg4/bin/mv will print a diagnostic message to stderr for all other failed
attempts to duplicate file characteristics. The exit code will not be affected.
In order to preserve the source file characteristics, users must have the appropriate file access permissions. This includes being super-user or having the same owner id as the destination file.

OPTIONS

The following options are supported:

- **-f** 
  `mv` will move the file(s) without prompting even if it is writing over an existing `target`. Note that this is the default if the standard input is not a terminal.

- **-i** 
  `mv` will prompt for confirmation whenever the move would overwrite an existing `target`. An affirmative answer means that the move should proceed. Any other answer prevents `mv` from overwriting the `target`.

**/usr/bin/mv**

Specifying both the `-f` and the `-i` options is not considered an error. The `-f` option will override the `-i` option.

**/usr/xpg4/bin/mv**

Specifying both the `-f` and the `-i` options is not considered an error. The last option specified will determine the behavior of `mv`.

OPERANDS

The following operands are supported:

- **source** 
  A path name of a file or directory to be moved.

- **target_file** 
  A new path name for the file or directory being moved.

- **target_dir** 
  A path name of an existing directory into which to move the input files.

USAGE

See `largefile(5)` for the description of the behavior of `mv` when encountering files greater than or equal to 2 Gbyte ($2^{31}$ bytes).

ENVIRONMENT VARIABLES

See `environ(5)` for descriptions of the following environment variables that affect the execution of `mv`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

EXIT STATUS

The following exit values are returned:

- **0** 
  All input files were moved successfully.

- **>0** 
  An error occurred.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>/usr/bin/mv</th>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
<td></td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
<td></td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Stable</td>
<td></td>
</tr>
</tbody>
</table>
mv(1)

/usr/xpg4/bin/mv

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWxcu4</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Standard</td>
</tr>
</tbody>
</table>

SEE ALSO

cp(1), cpio(1), ln(1), rm(1), setfacl(1), chmod(2), attributes(5), environ(5), fsattr(5), largefile(5), XPG4(5)

NOTES

A -- permits the user to mark explicitly the end of any command line options, allowing mv to recognize filename arguments that begin with a -. As an aid to BSD migration, mv will accept - as a synonym for --. This migration aid may disappear in a future release.
nawk – pattern scanning and processing language

/usr/bin/nawk [-F ERE] [-v assignment] ’program’ | -F progfile... [argument...]
/usr/xpg4/bin/awk [-F ERE] [-v assignment...] ’program’ | -F progfile... [argument...]

The /usr/bin/nawk and /usr/xpg4/bin/awk utilities execute programs written in the nawk programming language, which is specialized for textual data manipulation. A nawk program is a sequence of patterns and corresponding actions. The string specifying program must be enclosed in single quotes (’) to protect it from interpretation by the shell. The sequence of pattern-action statements can be specified in the command line as program or in one, or more, file(s) specified by the -f progfile option. When input is read that matches a pattern, the action associated with the pattern is performed.

Input is interpreted as a sequence of records. By default, a record is a line, but this can be changed by using the RS built-in variable. Each record of input is matched to each pattern in the program. For each pattern matched, the associated action is executed.

The nawk utility interprets each input record as a sequence of fields where, by default, a field is a string of non-blank characters. This default white-space field delimiter (blanks and/or tabs) can be changed by using the FS built-in variable or the -F ERE option. The nawk utility denotes the first field in a record $1, the second $2, and so forth. The symbol $0 refers to the entire record; setting any other field causes the reevaluation of $0. Assigning to $0 resets the values of all fields and the NF built-in variable.

The following options are supported:

- F ERE Define the input field separator to be the extended regular expression ERE, before any input is read (can be a character).
- F progfile Specifies the pathname of the file progfile containing a nawk program. If multiple instances of this option are specified, the concatenation of the files specified as progfile in the order specified is the nawk program. The nawk program can alternatively be specified in the command line as a single argument.
- v assignment The assignment argument must be in the same form as an assignment operand. The assignment is of the form var=value, where var is the name of one of the variables described below. The specified assignment occurs before executing the nawk program, including the actions associated with BEGIN patterns (if any). Multiple occurrences of this option can be specified.

The following operands are supported:

program If no -f option is specified, the first operand to nawk is the text of the nawk program. The application supplies the program operand.
as a single argument to *nawk*. If the text does not end in a newline character, *nawk* interprets the text as if it did.

**argument**

Either of the following two types of **argument** can be intermixed:

- **file**
  
  A pathname of a file that contains the input to be read, which is matched against the set of patterns in the program. If no **file** operands are specified, or if a **file** operand is -, the standard input is used.

- **assignment**
  
  An operand that begins with an underscore or alphabetic character from the portable character set, followed by a sequence of underscores, digits and alphabetics from the portable character set, followed by the = character specifies a variable assignment rather than a pathname. The characters before the = character represent the name of a *nawk* variable; if that name is a *nawk* reserved word the behavior is undefined. The characters following the equal sign are interpreted as if they appeared in the *nawk* program preceded and followed by a double-quote (" character, as a STRING token, except that if the last character is an unescaped backslash, it is interpreted as a literal backslash rather than as the first character of the sequence "\". The variable is assigned the value of that STRING token. If the value is considered a numeric string, the variable is assigned its numeric value. Each such variable assignment is performed just before the processing of the following **file**, if any. Thus, an assignment before the first **file** argument is executed after the **BEGIN** actions (if any), while an assignment after the last **file** argument is executed before the **END** actions (if any). If there are no **file** arguments, assignments are executed before processing the standard input.

**INPUT FILES**

Input files to the *nawk* program from any of the following sources:

- any **file** operands or their equivalents, achieved by modifying the *nawk* variables `ARGV` and `ARGC`
- standard input in the absence of any **file** operands
- arguments to the `getline` function
must be text files. Whether the variable RS is set to a value other than a newline character or not, for these files, implementations support records terminated with the specified separator up to \{LINE_MAX\} bytes and may support longer records.

If -f progfile is specified, the files named by each of the progfile option-arguments must be text files containing an awk program.

The standard input are used only if no file operands are specified, or if a file operand is -.

A awk program is composed of pairs of the form:

\[
\text{pattern} \{ \text{action} \}
\]

Either the pattern or the action (including the enclosing brace characters) can be omitted. Pattern-action statements are separated by a semicolon or by a newline.

A missing pattern matches any record of input, and a missing action is equivalent to an action that writes the matched record of input to standard output.

Execution of the awk program starts by first executing the actions associated with all BEGIN patterns in the order they occur in the program. Then each file operand (or standard input if no files were specified) is processed by reading data from the file until a record separator is seen (a newline character by default), splitting the current record into fields using the current value of FS, evaluating each pattern in the program in the order of occurrence, and executing the action associated with each pattern that matches the current record. The action for a matching pattern is executed before evaluating subsequent patterns. Last, the actions associated with all END patterns is executed in the order they occur in the program.

Expressions describe computations used in patterns and actions. In the following table, valid expression operations are given in groups from highest precedence first to lowest precedence last, with equal-precedence operators grouped between horizontal lines. In expression evaluation, where the grammar is formally ambiguous, higher precedence operators are evaluated before lower precedence operators. In this table expr, expr1, expr2, and expr3 represent any expression, while lvalue represents any entity that can be assigned to (that is, on the left side of an assignment operator).

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Name</th>
<th>Type of Result</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>( expr )</td>
<td>Grouping</td>
<td>type of expr</td>
<td>n/a</td>
</tr>
<tr>
<td>$expr</td>
<td>Field reference</td>
<td>string</td>
<td>n/a</td>
</tr>
<tr>
<td>++ lvalue</td>
<td>Pre-increment</td>
<td>numeric</td>
<td>n/a</td>
</tr>
<tr>
<td>-- lvalue</td>
<td>Pre-decrement</td>
<td>numeric</td>
<td>n/a</td>
</tr>
<tr>
<td>lvalue ++</td>
<td>Post-increment</td>
<td>numeric</td>
<td>n/a</td>
</tr>
</tbody>
</table>
### Syntax Name Type of Result Associativity

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Name</th>
<th>Type of Result</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>lvalue</code></td>
<td>Post-decrement</td>
<td>numeric</td>
<td>n/a</td>
</tr>
<tr>
<td><code>expr ^ expr</code></td>
<td>Exponentiation</td>
<td>numeric</td>
<td>right</td>
</tr>
<tr>
<td><code>! expr</code></td>
<td>Logical not</td>
<td>numeric</td>
<td>n/a</td>
</tr>
<tr>
<td><code>+ expr</code></td>
<td>Unary plus</td>
<td>numeric</td>
<td>n/a</td>
</tr>
<tr>
<td><code>− expr</code></td>
<td>Unary minus</td>
<td>numeric</td>
<td>n/a</td>
</tr>
<tr>
<td><code>expr * expr</code></td>
<td>Multiplication</td>
<td>numeric</td>
<td>left</td>
</tr>
<tr>
<td><code>expr / expr</code></td>
<td>Division</td>
<td>numeric</td>
<td>left</td>
</tr>
<tr>
<td><code>expr % expr</code></td>
<td>Modulus</td>
<td>numeric</td>
<td>left</td>
</tr>
<tr>
<td><code>expr + expr</code></td>
<td>Addition</td>
<td>numeric</td>
<td>left</td>
</tr>
<tr>
<td><code>expr − expr</code></td>
<td>Subtraction</td>
<td>numeric</td>
<td>left</td>
</tr>
<tr>
<td><code>expr expr</code></td>
<td>String concatenation</td>
<td>string</td>
<td>left</td>
</tr>
<tr>
<td><code>expr &lt; expr</code></td>
<td>Less than</td>
<td>numeric</td>
<td>none</td>
</tr>
<tr>
<td><code>expr &lt;= expr</code></td>
<td>Less than or equal to</td>
<td>numeric</td>
<td>none</td>
</tr>
<tr>
<td><code>expr != expr</code></td>
<td>Not equal to</td>
<td>numeric</td>
<td>none</td>
</tr>
<tr>
<td><code>expr == expr</code></td>
<td>Equal to</td>
<td>numeric</td>
<td>none</td>
</tr>
<tr>
<td><code>expr &gt; expr</code></td>
<td>Greater than</td>
<td>numeric</td>
<td>none</td>
</tr>
<tr>
<td><code>expr &gt;= expr</code></td>
<td>Greater than or equal to</td>
<td>numeric</td>
<td>none</td>
</tr>
<tr>
<td><code>expr ~ expr</code></td>
<td>ERE match</td>
<td>numeric</td>
<td>none</td>
</tr>
<tr>
<td><code>expr !~ expr</code></td>
<td>ERE non-match</td>
<td>numeric</td>
<td>none</td>
</tr>
<tr>
<td><code>expr in array</code></td>
<td>Array membership</td>
<td>numeric</td>
<td>left</td>
</tr>
<tr>
<td><code>( index ) in</code></td>
<td>Multi-dimension</td>
<td>numeric</td>
<td>left</td>
</tr>
<tr>
<td><code>expr</code></td>
<td>membership</td>
<td>numeric</td>
<td></td>
</tr>
<tr>
<td><code>@expr &amp; &amp; expr</code></td>
<td>Logical AND</td>
<td>numeric</td>
<td>left</td>
</tr>
<tr>
<td>`expr</td>
<td></td>
<td>expr`</td>
<td>Logical OR</td>
</tr>
<tr>
<td><code>expr1 ? expr2</code></td>
<td>Conditional expression</td>
<td>type of selected</td>
<td>right</td>
</tr>
<tr>
<td><code>: expr3</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>lvalue ^= expr</code></td>
<td>Exponentiation</td>
<td>numeric</td>
<td>right</td>
</tr>
<tr>
<td>Syntax</td>
<td>Name</td>
<td>Type of Result</td>
<td>Associativity</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>----------------</td>
<td>---------------</td>
</tr>
<tr>
<td>lvalue %= expr</td>
<td>Modulus assignment</td>
<td>numeric</td>
<td>right</td>
</tr>
<tr>
<td>lvalue *= expr</td>
<td>Multiplication</td>
<td>numeric</td>
<td>right</td>
</tr>
<tr>
<td>lvalue /= expr</td>
<td>Division assignment</td>
<td>numeric</td>
<td>right</td>
</tr>
<tr>
<td>lvalue += expr</td>
<td>Addition assignment</td>
<td>numeric</td>
<td>right</td>
</tr>
<tr>
<td>lvalue -= expr</td>
<td>Subtraction</td>
<td>numeric</td>
<td>right</td>
</tr>
<tr>
<td>lvalue = expr</td>
<td>Assignment</td>
<td>type of expr</td>
<td>right</td>
</tr>
</tbody>
</table>

Each expression has either a string value, a numeric value or both. Except as stated for specific contexts, the value of an expression is implicitly converted to the type needed for the context in which it is used. A string value is converted to a numeric value by the equivalent of the following calls:

```c
setlocale(LC_NUMERIC, "")
numeric_value = atof(string_value);
```

A numeric value that is exactly equal to the value of an integer is converted to a string by the equivalent of a call to the `sprintf` function with the string `%d` as the `fmt` argument and the numeric value being converted as the first and only `expr` argument. Any other numeric value is converted to a string by the equivalent of a call to the `sprintf` function with the value of the variable `CONVFMT` as the `fmt` argument and the numeric value being converted as the first and only `expr` argument.

A string value is considered to be a *numeric string* in the following case:

1. Any leading and trailing blank characters is ignored.
2. If the first unignored character is a `+` or `−`, it is ignored.
3. If the remaining unignored characters would be lexically recognized as a `NUMBER` token, the string is considered a *numeric string*.

If a `−` character is ignored in the above steps, the numeric value of the *numeric string* is the negation of the numeric value of the recognized `NUMBER` token. Otherwise the numeric value of the *numeric string* is the numeric value of the recognized `NUMBER` token. Whether or not a string is a *numeric string* is relevant only in contexts where that term is used in this section.

When an expression is used in a Boolean context, if it has a numeric value, a value of zero is treated as false and any other value is treated as true. Otherwise, a string value of the null string is treated as false and any other value is treated as true. A Boolean context is one of the following:
nawk(1)

- the first subexpression of a conditional expression.
- an expression operated on by logical NOT, logical AND, or logical OR.
- the second expression of a for statement.
- the expression of an if statement.
- the expression of the while clause in either a while or do . . . while statement.
- an expression used as a pattern (as in Overall Program Structure).

The nawk language supplies arrays that are used for storing numbers or strings. Arrays need not be declared. They are initially empty, and their sizes change dynamically. The subscripts, or element identifiers, are strings, providing a type of associative array capability. An array name followed by a subscript within square brackets can be used as an lvalue and as an expression, as described in the grammar. Unsubscripted array names are used in only the following contexts:

- a parameter in a function definition or function call.
- the NAME token following any use of the keyword in.

A valid array index consists of one or more comma-separated expressions, similar to the way in which multi-dimensional arrays are indexed in some programming languages. Because nawk arrays are really one-dimensional, such a comma-separated list is converted to a single string by concatenating the string values of the separate expressions, each separated from the other by the value of the SUBSEP variable.

Thus, the following two index operations are equivalent:

```
var[expr1, expr2, ... exprn]
var[expr1 SUBSEP expr2 SUBSEP ... SUBSEP exprn]
```

A multi-dimensioned index used with the in operator must be put in parentheses. The in operator, which tests for the existence of a particular array element, does not create the element if it does not exist. Any other reference to a non-existent array element automatically creates it.

Variables can be used in an nawk program by referencing them. With the exception of function parameters, they are not explicitly declared. Uninitialized scalar variables and array elements have both a numeric value of zero and a string value of the empty string.

Field variables are designated by a $ followed by a number or numerical expression. The effect of the field number expression evaluating to anything other than a non-negative integer is unspecified; uninitialized variables or string values need not be converted to numeric values in this context. New field variables are created by assigning a value to them. References to non-existent fields (that is, fields after $NF) produce the null string. However, assigning to a non-existent field (for example, $$(NF+2) = 5$$) increases the value of NF, create any intervening fields with the null string as their values and cause the value of $$0$$ to be recomputed, with the fields being separated by the value of OFS. Each field variable has a string value when created. If
the string, with any occurrence of the decimal-point character from the current locale changed to a period character, is considered a numeric string (see Expressions in nawk above), the field variable also has the numeric value of the numeric string.

nawk sets the following special variables:

ARGC
The number of elements in the ARGV array.

ARGV
An array of command line arguments, excluding options and the program argument, numbered from zero to ARGC−1.

The arguments in ARGV can be modified or added to; ARGC can be altered. As each input file ends, nawk treats the next non-null element of ARGV, up to the current value of ARGC−1, inclusive, as the name of the next input file. Setting an element of ARGV to null means that it is not treated as an input file. The name − indicates the standard input. If an argument matches the format of an assignment operand, this argument is treated as an assignment rather than a file argument.

CONVFMT
The printf format for converting numbers to strings (except for output statements, where OFMT is used); %.6g by default.

ENVIRON
The variable ENVIRON is an array representing the value of the environment. The indices of the array are strings consisting of the names of the environment variables, and the value of each array element is a string consisting of the value of that variable. If the value of an environment variable is considered a numeric string, the array element also has its numeric value.

In all cases where nawk behavior is affected by environment variables (including the environment of any commands that nawk executes via the system function or via pipeline redirections with the printf statement, the printf statement, or the getline function), the environment used is the environment at the time nawk began executing.

FILENAME
A pathname of the current input file. Inside a BEGIN action the value is undefined. Inside an END action the value is the name of the last input file processed.

FNR
The ordinal number of the current record in the current file. Inside a BEGIN action the value is zero. Inside an END action the value is the number of the last record processed in the last file processed.

FS
Input field separator regular expression; a space character by default.

NF
The number of fields in the current record. Inside a BEGIN action, the use of NF is undefined unless a getline function without a var argument is executed previously. Inside an END action, NF retains the value it had for the last record read, unless a

/usr/xpg4/bin/awk
subsequent, redirected, \texttt{getline} function without a \texttt{var} argument is performed prior to entering the \texttt{END} action.

\textbf{NR} \hspace{1cm} The ordinal number of the current record from the start of input. Inside a \texttt{BEGIN} action the value is zero. Inside an \texttt{END} action the value is the number of the last record processed.

\textbf{OFMT} \hspace{1cm} The \texttt{printf} format for converting numbers to strings in output statements "\texttt{\%6g}" by default. The result of the conversion is unspecified if the value of \texttt{OFMT} is not a floating-point format specification.

\textbf{OFS} \hspace{1cm} The \texttt{printf} statement output field separator; a space character by default.

\textbf{ORS} \hspace{1cm} The \texttt{printf} output record separator; a newline character by default.

\textbf{LENGTH} \hspace{1cm} The length of the string matched by the \texttt{match} function.

\textbf{RS} \hspace{1cm} The first character of the string value of \texttt{RS} is the input record separator; a newline character by default. If \texttt{RS} contains more than one character, the results are unspecified. If \texttt{RS} is null, then records are separated by sequences of one or more blank lines: leading or trailing blank lines do not produce empty records at the beginning or end of input, and the field separator is always newline, no matter what the value of \texttt{FS}.

\textbf{RSTART} \hspace{1cm} The starting position of the string matched by the \texttt{match} function, numbering from 1. This is always equivalent to the return value of the \texttt{match} function.

\textbf{SUBSEP} \hspace{1cm} The subscript separator string for multi-dimensional arrays; the default value is \texttt{1}

The \texttt{nawk} utility makes use of the extended regular expression notation (see \texttt{regex(5)}) except that it allows the use of C-language conventions to escape special characters within the EREs, namely \texttt{\_, \a, \b, \f, \n, \r, \t, \v}, and those specified in the following table. These escape sequences are recognized both inside and outside bracket expressions. Note that records need not be separated by newline characters and string constants can contain newline characters, so even the \texttt{\n} sequence is valid in \texttt{nawk} EREs. Using a slash character within the regular expression requires escaping as shown in the table below:

<table>
<thead>
<tr>
<th>Escape Sequence</th>
<th>Description</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{&quot;}</td>
<td>Backslash quotation-mark</td>
<td>Quotation-mark character</td>
</tr>
<tr>
<td>\texttt{/}</td>
<td>Backslash slash</td>
<td>Slash character</td>
</tr>
</tbody>
</table>
A backslash character followed by the longest sequence of one, two, or three octal-digit characters (01234567). If all of the digits are 0, (that is, representation of the NULL character), the behavior is undefined.

The character encoded by the one-, two- or three-digit octal integer. Multi-byte characters require multiple, concatenated escape sequences, including the leading \ for each byte.

A backslash character followed by any character not described in this table or special characters (\ \a \b \f \n \r \t \v).

Undefined

A regular expression can be matched against a specific field or string by using one of the two regular expression matching operators, - and ! -. These operators interpret their right-hand operand as a regular expression and their left-hand operand as a string. If the regular expression matches the string, the - expression evaluates to the value 1, and the ! - expression evaluates to the value 0. If the regular expression does not match the string, the - expression evaluates to the value 0, and the ! - expression evaluates to the value 1. If the right-hand operand is any expression other than the lexical token ERE, the string value of the expression is interpreted as an extended regular expression, including the escape conventions described above. Note that these same escape conventions also are applied in the determining the value of a string literal (the lexical token STRING), and is applied a second time when a string literal is used in this context.

When an ERE token appears as an expression in any context other than as the right-hand of the - or ! - operator or as one of the built-in function arguments described below, the value of the resulting expression is the equivalent of:

$0 ~ /ere/

The ere argument to the gsub, match, sub functions, and the fs argument to the split function (see String Functions) is interpreted as extended regular expressions. These can be either ERE tokens or arbitrary expressions, and are interpreted in the same manner as the right-hand side of the - or ! - operator.

An extended regular expression can be used to separate fields by using the -F ERE option or by assigning a string containing the expression to the built-in variable FS. The default value of the FS variable is a single space character. The following describes FS behavior:

1. If FS is a single character:
   - If FS is the space character, skip leading and trailing blank characters; fields are delimited by sets of one or more blank characters.
   - Otherwise, if FS is any other character c, fields are delimited by each single occurrence of c.
2. Otherwise, the string value of \texttt{FS} is considered to be an extended regular expression. Each occurrence of a sequence matching the extended regular expression delimits fields.

Except in the \texttt{gsub}, \texttt{match}, \texttt{split}, and \texttt{sub} built-in functions, regular expression matching is based on input records; that is, record separator characters (the first character of the value of the variable \texttt{RS}, a newline character by default) cannot be embedded in the expression, and no expression matches the record separator character. If the record separator is not a newline character, newline characters embedded in the expression can be matched. In those four built-in functions, regular expression matching are based on text strings. So, any character (including the newline character and the record separator) can be embedded in the pattern and an appropriate pattern will match any character. However, in all \texttt{nawk} regular expression matching, the use of one or more NUL characters in the pattern, input record or text string produces undefined results.

A pattern is any valid expression, a range specified by two expressions separated by comma, or one of the two special patterns \texttt{BEGIN} or \texttt{END}.

The \texttt{nawk} utility recognizes two special patterns, \texttt{BEGIN} and \texttt{END}. Each \texttt{BEGIN} pattern is matched once and its associated action executed before the first record of input is read (except possibly by use of the \texttt{getline} function in a prior \texttt{BEGIN} action) and before command line assignment is done. Each \texttt{END} pattern is matched once and its associated action executed after the last record of input has been read. These two patterns have associated actions.

\texttt{BEGIN} and \texttt{END} do not combine with other patterns. Multiple \texttt{BEGIN} and \texttt{END} patterns are allowed. The actions associated with the \texttt{BEGIN} patterns are executed in the order specified in the program, as are the \texttt{END} actions. An \texttt{END} pattern can precede a \texttt{BEGIN} pattern in a program.

If an \texttt{nawk} program consists of only actions with the pattern \texttt{BEGIN}, and the \texttt{BEGIN} action contains no \texttt{getline} function, \texttt{nawk} exits without reading its input when the last statement in the last \texttt{BEGIN} action is executed. If an \texttt{nawk} program consists of only actions with the pattern \texttt{END} or only actions with the patterns \texttt{BEGIN} and \texttt{END}, the input is read before the statements in the \texttt{END} actions are executed.

An expression pattern is evaluated as if it were an expression in a Boolean context. If the result is true, the pattern is considered to match, and the associated action (if any) is executed. If the result is false, the action is not executed.

A pattern range consists of two expressions separated by a comma. In this case, the action is performed for all records between a match of the first expression and the following match of the second expression, inclusive. At this point, the pattern range can be repeated starting at input records subsequent to the end of the matched range.

An action is a sequence of statements. A statement may be one of the following:

\begin{verbatim}
if ( expression ) statement [ else statement ]
while ( expression ) statement
\end{verbatim}
do statement while ( expression )
for ( expression ; expression ; expression ) statement
for ( var in array ) statement
delete array[subscript] #delete an array element
break
continue
{ [ statement ] . . . }
expression       # commonly variable = expression
print [ expression-list ] [ >expression ]
printf format [ ,expression-list ] [ >expression ]
next             # skip remaining patterns on this input line
exit [expr]     # skip the rest of the input; exit status is expr
return [expr]

Any single statement can be replaced by a statement list enclosed in braces. The statements are terminated by newline characters or semicolons, and are executed sequentially in the order that they appear.

The next statement causes all further processing of the current input record to be abandoned. The behavior is undefined if a next statement appears or is invoked in a BEGIN or END action.

The exit statement invokes all END actions in the order in which they occur in the program source and then terminate the program without reading further input. An exit statement inside an END action terminates the program without further execution of END actions. If an expression is specified in an exit statement, its numeric value is the exit status of nawk, unless subsequent errors are encountered or a subsequent exit statement with an expression is executed.

Output Statements

Both print and printf statements write to standard output by default. The output is written to the location specified by output redirection if one is supplied, as follows:

> expression>> expression| expression

In all cases, the expression is evaluated to produce a string that is used as a full pathname to write into (for > or >>) or as a command to be executed (for |). Using the first two forms, if the file of that name is not currently open, it is opened, creating it if necessary and using the first form, truncating the file. The output then is appended to the file. As long as the file remains open, subsequent calls in which expression evaluates to the same string value simply appends output to the file. The file remains open until the close function, which is called with an expression that evaluates to the same string value.

The third form writes output onto a stream piped to the input of a command. The stream is created if no stream is currently open with the value of expression as its command name. The stream created is equivalent to one created by a call to the popen(3C) function with the value of expression as the command argument and a value of w as the mode argument. As long as the stream remains open, subsequent calls in which expression evaluates to the same string value writes output to the existing stream. The stream will remain open until the close function is called with an expression that evaluates to the same string value. At that time, the stream is closed as if by a call to the pclose function.
These output statements take a comma-separated list of expression s referred in the grammar by the non-terminal symbols expr_list, print_expr_list or print_expr_list_opt. This list is referred to here as the expression list, and each member is referred to as an expression argument.

The print statement writes the value of each expression argument onto the indicated output stream separated by the current output field separator (see variable OFFS above), and terminated by the output record separator (see variable ORS above). All expression arguments is taken as strings, being converted if necessary; with the exception that the printf format in OFMT is used instead of the value in CONVFMT. An empty expression list stands for the whole input record ($0).

The printf statement produces output based on a notation similar to the File Format Notation used to describe file formats in this document. Output is produced as specified with the first expression argument as the string format and subsequent expression arguments as the strings arg1 to argn, inclusive, with the following exceptions:

1. The format is an actual character string rather than a graphical representation. Therefore, it cannot contain empty character positions. The space character in the format string, in any context other than a flag of a conversion specification, is treated as an ordinary character that is copied to the output.
2. If the character set contains a Delta character and that character appears in the format string, it is treated as an ordinary character that is copied to the output.
3. The escape sequences beginning with a backslash character is treated as sequences of ordinary characters that are copied to the output. Note that these same sequences is interpreted lexically by awk when they appear in literal strings, but they is not treated specially by the printf statement.
4. A field width or precision can be specified as the * character instead of a digit string. In this case the next argument from the expression list is fetched and its numeric value taken as the field width or precision.
5. The implementation does not precede or follow output from the d or u conversion specifications with blank characters not specified by the format string.
6. The implementation does not precede output from the o conversion specification with leading zeros not specified by the format string.
7. For the c conversion specification: if the argument has a numeric value, the character whose encoding is that value is output. If the value is zero or is not the encoding of any character in the character set, the behavior is undefined. If the argument does not have a numeric value, the first character of the string value will be output; if the string does not contain any characters the behavior is undefined.
8. For each conversion specification that consumes an argument, the next expression argument will be evaluated. With the exception of the c conversion, the value will be converted to the appropriate type for the conversion specification.
9. If there are insufficient expression arguments to satisfy all the conversion specifications in the format string, the behavior is undefined.
10. If any character sequence in the format string begins with a % character, but does not form a valid conversion specification, the behavior is unspecified.

Both printf and printf can output at least \{LINE_MAX\} bytes.

The awk language has a variety of built-in functions: arithmetic, string, input/output and general.

The arithmetic functions, except for int, are based on the ISO C standard. The behavior is undefined in cases where the ISO C standard specifies that an error be returned or that the behavior is undefined. Although the grammar permits built-in functions to appear with no arguments or parentheses, unless the argument or parentheses are indicated as optional in the following list (by displaying them within the [ ] brackets), such use is undefined.

atan2(y,x) Return arctangent of \( y/x \).

\( \text{cos}(x) \) Return cosine of \( x \), where \( x \) is in radians.

\( \text{sin}(x) \) Return sine of \( x \), where \( x \) is in radians.

\( \text{exp}(x) \) Return the exponential function of \( x \).

\( \text{log}(x) \) Return the natural logarithm of \( x \).

\( \text{sqrt}(x) \) Return the square root of \( x \).

\( \text{int}(x) \) Truncate its argument to an integer. It will be truncated toward 0 when \( x > 0 \).

\( \text{rand}() \) Return a random number \( n \), such that \( 0 \leq n < 1 \).

\( \text{srand}([\text{expr}]) \) Set the seed value for rand to \( \text{expr} \) or use the time of day if \( \text{expr} \) is omitted. The previous seed value will be returned.

The string functions in the following list shall be supported. Although the grammar permits built-in functions to appear with no arguments or parentheses, unless the argument or parentheses are indicated as optional in the following list (by displaying them within the [ ] brackets), such use is undefined.

\( \text{gsub}(\text{ere},\text{repl},\text{in}) \) Behave like sub (see below), except that it will replace all occurrences of the regular expression (like the ed utility global substitute) in \( \$0 \) or in the in argument, when specified.

\( \text{index}(s,t) \) Return the position, in characters, numbering from 1, in string \( s \) where string \( t \) first occurs, or zero if it does not occur at all.

\( \text{length}([s]) \) Return the length, in characters, of its argument taken as a string, or of the whole record, \( \$0 \), if there is no argument.
match(s,ere)

Return the position, in characters, numbering from 1, in string s where the extended regular expression ere occurs, or zero if it does not occur at all. RSTART will be set to the starting position (which is the same as the returned value), zero if no match is found; RLENGTH will be set to the length of the matched string, −1 if no match is found.

split(s,a[,,fs])

Split the string s into array elements a[1], a[2], ... , a[n], and return n. The separation will be done with the extended regular expression fs or with the field separator FS if fs is not given. Each array element will have a string value when created. If the string assigned to any array element, with any occurrence of the decimal-point character from the current locale changed to a period character, would be considered a numeric string; the array element will also have the numeric value of the numeric string. The effect of a null string as the value of fs is unspecified.

sprintf(fmt,expr,expr,...)

Format the expressions according to the printf format given by fmt and return the resulting string.

sub(ere,repl[,in])

Substitute the string repl in place of the first instance of the extended regular expression ERE in string in and return the number of substitutions. An ampersand (&) appearing in the string repl will be replaced by the string from in that matches the regular expression. For each occurrence of backslash (\) encountered when scanning the string repl from beginning to end, the next character is taken literally and loses its special meaning (for example, \& will be interpreted as a literal ampersand character). Except for & and \, it is unspecified what the special meaning of any such character is. If in is specified and it is not an lvalue the behavior is undefined. If in is omitted, nawk will substitute in the current record ($0) .
nawk(1)

**Input/Output and General Functions**

### close(expression)
Close the file or pipe opened by a `printf` statement or a call to `getline` with the same string-valued expression. If the close was successful, the function will return 0; otherwise, it will return non-zero.

### expression | getline[var]
Read a record of input from a stream piped from the output of a command. The stream will be created if no stream is currently open with the value of `expression` as its command name. The stream created will be equivalent to one created by a call to the `popen` function with the value of `expression` as the `command` argument and a value of `r` as the `mode` argument. As long as the stream remains open, subsequent calls in which `expression` evaluates to the same string value will read subsequent records from the file. The stream will remain open until the `close` function is called with an expression that evaluates to the same string value. At
that time, the stream will be closed as if by a call to the pclose function. If var is missing, $0 and NF will be set; otherwise, var will be set.

The getline operator can form ambiguous constructs when there are operators that are not in parentheses (including concatenate) to the left of the | (to the beginning of the expression containing getline). In the context of the $ operator, | behaves as if it had a lower precedence than $. The result of evaluating other operators is unspecified, and all such uses of portable applications must be put in parentheses properly.

getline
Set $0 to the next input record from the current input file. This form of getline will set the NF, NR, and FNR variables.

ggetline var
Set variable var to the next input record from the current input file. This form of getline will set the FNR and NR variables.

ggetline [var] < expression
Read the next record of input from a named file. The expression will be evaluated to produce a string that is used as a full pathname. If the file of that name is not currently open, it will be opened. As long as the stream remains open, subsequent calls in which expression evaluates to the same string value will read subsequent records from the file. The file will remain open until the close function is called with an expression that evaluates to the same string value. If var is missing, $0 and NF will be set; otherwise, var will be set.

The getline operator can form ambiguous constructs when there are binary operators that are not in parentheses (including concatenate) to the right of the < (up to the end of the expression containing the getline). The result of evaluating such a construct is unspecified, and all such uses of portable applications must be put in parentheses properly.
system(expression)  
Execute the command given by expression in a manner equivalent to the system(3C) function and return the exit status of the command.

All forms of getline will return 1 for successful input, 0 for end of file, and −1 for an error.

Where strings are used as the name of a file or pipeline, the strings must be textually identical. The terminology “same string value” implies that “equivalent strings”, even those that differ only by space characters, represent different files.

The nawk language also provides user-defined functions. Such functions can be defined as:

```
funtion name(args, . . .) { statements }  
```

A function can be referred to anywhere in an nawk program; in particular, its use can precede its definition. The scope of a function will be global.

Function arguments can be either scalars or arrays; the behavior is undefined if an array name is passed as an argument that the function uses as a scalar, or if a scalar expression is passed as an argument that the function uses as an array. Function arguments will be passed by value if scalar and by reference if array name. Argument names will be local to the function; all other variable names will be global. The same name will not be used as both an argument name and as the name of a function or a special nawk variable. The same name must not be used both as a variable name with global scope and as the name of a function. The same name must not be used within the same scope both as a scalar variable and as an array.

The number of parameters in the function definition need not match the number of parameters in the function call. Excess formal parameters can be used as local variables. If fewer arguments are supplied in a function call than are in the function definition, the extra parameters that are used in the function body as scalars will be initialized with a string value of the null string and a numeric value of zero, and the extra parameters that are used in the function body as arrays will be initialized as empty arrays. If more arguments are supplied in a function call than are in the function definition, the behavior is undefined.

When invoking a function, no white space can be placed between the function name and the opening parenthesis. Function calls can be nested and recursive calls can be made upon functions. Upon return from any nested or recursive function call, the values of all of the calling function’s parameters will be unchanged, except for array parameters passed by reference. The return statement can be used to return a value. If a return statement appears outside of a function definition, the behavior is undefined.
In the function definition, newline characters are optional before the opening brace and after the closing brace. Function definitions can appear anywhere in the program where a pattern-action pair is allowed.

**USAGE**

The `index`, `length`, `match`, and `substr` functions should not be confused with similar functions in the ISO C standard; the `nawk` versions deal with characters, while the ISO C standard deals with bytes.

Because the concatenation operation is represented by adjacent expressions rather than an explicit operator, it is often necessary to use parentheses to enforce the proper evaluation precedence.

See `largefile(5)` for the description of the behavior of `nawk` when encountering files greater than or equal to 2 Gbyte (2^31 bytes).

**EXAMPLES**

The `nawk` program specified in the command line is most easily specified within single-quotes (for example, `'program'`) for applications using `sh`, because `nawk` programs commonly contain characters that are special to the shell, including double-quotes. In the cases where a `nawk` program contains single-quote characters, it is usually easiest to specify most of the program as strings within single-quotes concatenated by the shell with quoted single-quote characters. For example:

```bash
awk '/\'\'/ { print "quote:\", $0 }' prints all lines from the standard input containing a single-quote character, prefixed with `quote:`.
```

The following are examples of simple `nawk` programs:

**EXAMPLE 1** Write to the standard output all input lines for which field 3 is greater than 5:

```
$3 > 5
```

**EXAMPLE 2** Write every tenth line:

```
(NR % 10) == 0
```

**EXAMPLE 3** Write any line with a substring matching the regular expression:

```
/(G|D)([0-9][[:alpha:]]+)/
```

**EXAMPLE 4** Print any line with a substring containing a G or D, followed by a sequence of digits and characters:

This example uses character classes `digit` and `alpha` to match language-independent digit and alphabetic characters, respectively.

```
/(G|D)([[:digit:]][[:alpha:]])*/
```
EXAMPLE 4 Print any line with a substring containing a G or D, followed by a sequence of digits and characters: (Continued)

EXAMPLE 5 Write any line in which the second field matches the regular expression and the fourth field does not:
```
$2 ~ /xyz/ & & $4 !~ /xyz/
```

EXAMPLE 6 Write any line in which the second field contains a backslash:
```
$2 ~ /\/
```

EXAMPLE 7 Write any line in which the second field contains a backslash (alternate method):
Note that backslash escapes are interpreted twice, once in lexical processing of the string and once in processing the regular expression.
```
$2 ~ "\\\"
```

EXAMPLE 8 Write the second to the last and the last field in each line, separating the fields by a colon:
```
{OFS=":";print $(NF-1), $NF}
```

EXAMPLE 9 Write the line number and number of fields in each line:
The three strings representing the line number, the colon and the number of fields are concatenated and that string is written to standard output.
```
{print NR":"NF}
```

EXAMPLE 10 Write lines longer than 72 characters:
```
{length($0) > 72}
```

EXAMPLE 11 Write first two fields in opposite order separated by the OFS:
```
{ print $2, $1 }
```

EXAMPLE 12 Same, with input fields separated by comma or space and tab characters, or both:
```
BEGIN { FS = ",,\[t\]*|\[t\]+* } 
{ print $2, $1 }
```

EXAMPLE 13 Add up first column, print sum and average:
```
{s += $1 }
END {print "sum is ", s, " average is", s/NR}
```
EXAMPLE 13 Add up first column, print sum and average: (Continued)

EXAMPLE 14 Write fields in reverse order, one per line (many lines out for each line in):
{ for (i = NF; i > 0; --i) print $i }

EXAMPLE 15 Write all lines between occurrences of the strings “start” and “stop”:
/start/, /stop/

EXAMPLE 16 Write all lines whose first field is different from the previous one:
$1 != prev { print; prev = $1 }

EXAMPLE 17 Simulate the echo command:
BEGIN {
 for (i = 1; i < ARGC; ++i)
   printf "%s%s", ARGV[i], i==ARGC-1?"\n":"
 }

EXAMPLE 18 Write the path prefixes contained in the PATH environment variable, one per line:
BEGIN {
   n = split (ENVIRON["PATH"], path, ":")
   for (i = 1; i <= n; ++i)
     print path[i]
 }

EXAMPLE 19 Print the file “input”, filling in page numbers starting at 5:
If there is a file named input containing page headers of the form
Page#

and a file named program that contains
/Page/ { $2 = n++; }
{ print }

then the command line
awk -f program n=5 input

will print the file input, filling in page numbers starting at 5.

ENVIRONMENT VARIABLES
See environ(5) for descriptions of the following environment variables that affect execution: LC_COLLATE, LC_CTYPE, LC_MESSAGES, and NLSPATH.

LC_NUMERIC Determine the radix character used when interpreting numeric input, performing conversions between numeric and string values and formatting numeric output. Regardless of locale, the period
character (the decimal-point character of the POSIX locale) is the
decimal-point character recognized in processing awk programs
(including assignments in command-line arguments).

EXIT STATUS

The following exit values are returned:

0  All input files were processed successfully.
>0  An error occurred.

The exit status can be altered within the program by using an exit expression.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>/usr/bin/nawk</th>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>/usr/xpg4/bin/awk</th>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWxcu4</td>
<td></td>
</tr>
</tbody>
</table>

SEE ALSO

awk(1), ed(1), egrep(1), grep(1), lex(1), sed(1), popen (3C), printf(3C),
system(3C), attributes(5), environ(5), largefile(5), regex(5), XPG4(5)

Aho, A. V., B. W. Kernighan, and P. J. Weinberger, The AWK Programming Language,

DIAGNOSTICS

If any file operand is specified and the named file cannot be accessed, nawk will write
a diagnostic message to standard error and terminate without any further action.

If the program specified by either the program operand or a proglfile operand is not a
valid nawk program (as specified in EXTENDED DESCRIPTION), the behavior is
undefined.

NOTES

Input white space is not preserved on output if fields are involved.

There are no explicit conversions between numbers and strings. To force an expression
to be treated as a number add 0 to it; to force it to be treated as a string concatenate the
null string (""") to it.
nca(1)

NAME
nca, snca – the Solaris Network Cache and Accelerator (NCA)

DESCRIPTION
The Solaris Network Cache and Accelerator ("NCA") is a kernel module designed to
provide improved web server performance. The kernel module, ncakmod, services
HTTP requests. To improve the performance of servicing HTTP requests, the NCA
kernel module maintains an in-kernel cache of web pages. If the NCA kernel module
cannot service the request itself, it passes the request to the http daemon (httpd). It
uses either a sockets interface, with family type designated PF_NCA, or a private
Solaris doors interface that is based on the Solaris doors RPC mechanism, to pass the
request.

To use the sockets interface, the web server must open a socket of family type PF_NCA.
The PF_NCA family supports only SOCK_STREAM and protocol 0, otherwise an error
occurs.

The following features are not presently supported:

■ You cannot initiate a connection from a PF_NCA type socket. The
  connect(3SOCKET) interface on PF_NCA will fail.

■ System calls that are associated with type SO_DGRAM, such as send(), sendto(),
  sendmsg(), recv(), recvfrom(), and recvmsg(), will fail.

■ You cannot set TCP or IP options on a PF_NCA type socket through
  setsockopt(3SOCKET).

The NCA cache consistency is maintained by honoring HTTP headers that deal with a
given content type and expiration date, much the same way as a proxy cache.

For configuration information, see System Administration Guide, Volume 3

When native PF_NCA socket support does not exist in the web server, the
ncad_addr(4) interface must be used to provide NCA support in that web server.

NCA is intended to be run on a dedicated web server. Running other large processes
while running NCA might cause undesirable behavior.

NCA supports the logging of in-kernel cache hits. See ncalogd.conf(4). NCA stores
logs in a binary format. Use the ncab2clf(1) utility to convert the log from a binary
format to the Common Log File format.

FILES
/etc/nca/ncakmod.conf  Lists configuration parameters for NCA.
/etc/nca/ncalogd.conf  Lists configuration parameters for NCA
                      logging.
/etc/nca/nca.if       Lists the physical interfaces on which NCA
                      will run.
/etc/hostname.{{0-9}}  Lists all physical interfaces configured on
                      the server.


ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWncar (32-bit)</td>
</tr>
<tr>
<td></td>
<td>SUNWncarx (64-bit)</td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Evolving</td>
</tr>
</tbody>
</table>

SEE ALSO

ncab2clf(1), ncakmod(1), close(2), read(2), write(2), sendfilev(3EXT), accept(3SOCKET), bind(3SOCKET) connect(3SOCKET), door_bind(3DOOR), door_call(3DOOR), door_create(3DOOR), getsockopt(3SOCKET), listen(3SOCKET), setsockopt(3SOCKET), shutdown(3SOCKET), socket(3HEAD), socket(3SOCKET), ncad_addr(4), nca.if(4), ncakmod.conf(4), ncalogd.conf(4), attributes(5)

System Administration Guide, Volume 3
**ncab2clf(1)**

**NAME**
ncab2clf – convert binary log file to Common Log File format

**SYNOPSIS**
/usr/bin/ncab2clf [-Dhv] [-i input-file] [-o output-file] [-b size] [-n number] [-s datetime]

**DESCRIPTION**
The `ncab2clf` command is used to convert the log file generated by the Solaris Network Cache and Accelerator (“NCA”) from binary format, to Common Log File (“CLF”) format. If no `input-file` is specified, `ncab2clf` uses stdin. If no `output-file` is specified, the output goes to stdout.

**OPTIONS**
- **-b**
  Specifies the binary-log-file blocking in kilobytes; the default is 64 Kbyte.
- **-D**
  Specifies that direct I/O be disabled.
- **-h**
  Prints usage message.
- **-i input-file**
  Specifies the input file.
- **-n number**
  Output number CLF records.
- **-o output-file**
  Specifies the output file.
- **-s datetime**
  Skip any records before the date and time specified in `datetime`. You can specify the date and time in CLF format or in the format specified by the `touch(1)` utility. CLF format is the dominant format, so `ncab2clf` first analyzes `datetime` assuming CLF.
- **-v**
  Provides verbose output.

**EXAMPLES**

**EXAMPLE 1** Converting a Binary File to a Common Log File Format

The following example converts the binary file `/var/nca/logs/nca.blf` to a file `/var/nca/logs/nca.clf`, which is in Common Log File format.

```
example% ncab2clf -D -i /var/nca/logs/nca.blf -o /var/nca/logs/nca.clf
```

**EXAMPLE 2** Converting Multiple Log Files

The following script may be used to convert multiple log files. The directory designated by "*" must only contain log files.

```
#!/bin/ksh
for filename in *
do
  ncab2clf -D < $filename > $filename.clf
done
```

**EXAMPLE 3** Using `-s` and `-n` on a Raw Device

The following example shows how `ncab2clf` can be used on a raw device. If not using the `-n` option, the default is to convert all records from the starting location to the end of the file. The date and time specified with `-s`, below, is in CLF format.
EXAMPLE 3 Using -s and -n on a Raw Device  (Continued)

example$ ncab2clf -s '10/Apr/2001:09:23:13' -n 100 < /dev/dsk/c2t1d0s6

EXIT STATUS  The following exit values are returned:
  0      The file converted successfully
 >0     An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWncau</td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Evolving</td>
</tr>
</tbody>
</table>

SEE ALSO  nca(1), ncakmod(1), nca.if(4), ncakmod.conf(4), ncalgd.conf(4), attributes(5)

System Administration Guide, Volume 3

NOTES  The binary log files generated by NCA can become very large. When converting these
large binary files, use the -b option to the ncab2clf command to help performance.

Direct I/O is a benefit to the user if the data being written does not come in as large
chunks. However, if the user wishes to convert the log file in large chunks using the
-b option, then direct I/O should be disabled by using the -D option.
ncakmod(1)

NAME
ncakmod – start or stop the NCA kernel module

SYNOPSIS
/etc/init.d/ncakmod start | stop

DESCRIPTION
ncakmod is used to start or stop the Solaris Network Cache and Accelerator (“NCA”) kernel module.

When the start option is specified at the command-line, the NCA kernel module will be activated for all physical interfaces listed in the nca.if file. When the ncakmod command is invoked with the stop option, the NCA kernel module will print the following message:

To stop NCA, please set the status configuration parameter to disable in ncakmod.conf and then reboot your system. See the ncakmod.conf(4) manual page for more information.

Note that in order to properly stop NCA on your system, you must first edit the ncakmod.conf(4) file and set the status field to “disable,” then reboot your system.

OPTIONS
   start     Starts the NCA kernel module.
   stop      Describes the current method for stopping the NCA feature.

EXAMPLES
EXAMPLE 1 Starting and Stopping the NCA Feature

The following command is used to start the NCA feature:

example$ /etc/init.d/ncakmod start

FILES
/etc/init.d/ncakmod        The NCA kernel module startup script.
/etc/nca/ncakmod.conf      Specifies configuration options for the NCA kernel module.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWncar</td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Evolving</td>
</tr>
</tbody>
</table>

SEE ALSO
nca(1), ncab2clf(1), ncad_addr(4), nca.if(4), ncakmod.conf(4), ncalogd.conf(4), attributes(5)
### NAME
netscape – start Netscape Communicator for Solaris

### SYNOPSIS
/usr/dt/bin/netscape [options] [arguments]

### DESCRIPTION
Netscape Communicator for Solaris is comprised of a comprehensive set of components that integrates browsing, email, web-based word processing, chat, and group scheduling to allow users to easily communicate, share, and access information.

### OPTIONS
Any argument that is not a switch is interpreted as either a file or URL. The following options are supported:

- `-component-bar`
  Shows only the Component Bar.

- `-composer`
  Opens all command line URLs in Composer.

- `-discussions`
  Shows Collabra Discussions.

- `-display dpy`
  Specifies the X server to use for display.

- `-dont-force-window-stacking`
  Ignores the alwaysraised, alwayslowered, and z-lock JavaScript `window.open()` attributes.

- `-dont-save-geometry-prefs`
  Does not save window geometry preferences for the session.

- `-edit`
  See `-composer`.

- `-geometry =WxH+X+Y`
  Positions and sizes the Netscape window.

- `-help`
  Shows the command line options for Netscape.

- `-iconic`
  Minimizes Netscape after start up.

- `-id window-id`
  Identifies an X window to receive `-remote` commands. If you do not specify a window, the first window found is used.

- `-ignore-geometry-prefs`
  Ignores saved window geometry preferences for the current session.

- `-install`
  Installs private colormap.
### -irix-session-management
Enables IRIX session management. On SGI systems, IRIX session management is enabled by default. IRIX session management is available on other platforms and may work with session managers other than the IRIX desktop. See -no-irix-session-management.

### -mail
Same as -messenger.

### -messenger
Shows the Messenger Mailbox (Inbox).

### -mono
Forces a one-bit deep image display.

### -ncols N
Sets the maximum number of colors to allocate for images when not using -install.

### -nethelp
Starts NetHelp, Netscape’s online help system.

### -news
Same as -discussions.

### -no-about-splash
Bypasses the startup license page.

### -no-install
Uses the default colormap.

### -no-irix-session-management
Disables IRIX session management. See -irix-session-management.

### -no-session-management
Disables session management. Session management is enabled by default. See -session-management.

### -noraise
Does not display the remote window on top when using -remote commands. See -raise and -remote.

### -raise
Displays the remote window on top when using -remote commands. See -noraise and -remote.

### -remote remote-command
Connects to and controls an existing process. You can issue multiple -remote options on the same command line. The commands are executed sequentially unless a command fails. If there is no Netscape process currently running, this command fails. If the command fails, an error message will be reported to stderr and it will exit with a nonzero status. See REMOTE ACTIONS section below and EXAMPLES.
The following options exist for finer-grained control of the -remote commands:

- **-id X_window_ID** If there is more than one Netscape Navigator window open, this option selects the window to control. If you do not use this option, the first window found is controlled. See EXAMPLES.

- **-raise**
- **-noraise** Controls whether the -remote command causes the Netscape window to raise/not raise itself to the top. The default is raise.

You can use -raise and -noraise options with the addBookmark and openURL arguments. See EXAMPLES.

- **-session-management** Enables session management. Session management is enabled by default. See -no-session-management.

- **-version** Displays the version number and build date.

- **-visual id-or-number** Uses the specified server visual.

- **-xrm resource-spec** Sets the specified X resource.

When Netscape Navigator is invoked with the -remote argument, it does not open a window, but instead connects to and controls an already existing process. The argument to the -remote switch is an Xt action to invoke, with optional arguments. Remote control is implemented using X properties, so the two processes need not be running on the same machine, and need not share a file system. See http://home.netscape.com/newsref/std/x-remote.html.

All of Netscape’s action names are the same as its resource names. For example, if you wanted to know the name of the action that corresponds to the “Add Bookmark” menu item, you could look in Netscape for “Add Bookmark” and see that the resource that is set to that string is addBookmark. That is the name of the Action as well. Note: To find the Netscape file, use the full path name which is, by default, /usr/dt/appconfig/netscape/lib/locale/C/app-defaults/Netscape.

You can use Actions in Translation tables in the usual Xt manner, but you can also invoke them directly via the -remote option, like this:

```bash
netscape -remote 'addBookmark()' That command will cause the existing Netscape Navigator process to add its current URL to the bookmarks, just as if you had selected that menu item.
```

To open a document, enter:
netscape(1)

```bash
netscape -remote 'openURL(http://home.netscape.com)'
```

Invoking an action with no arguments has the same effect as selecting the corresponding menu item. However, with some actions you can pass the following arguments:

- `addBookmark()`: Adds the current document to the Bookmark list.
- `addBookmark(URL)`: Adds the specified document to the Bookmark list. See EXAMPLES.
- `addBookmark(URL, title)`: Adds the specified document and title to the Bookmark list.
- `mailto()`: Opens the mail dialog box with an empty To: field.
- `mailto(a, b, c)`: Inserts the specified address(es) in the default To: field.
- `openFile()`: Opens a dialog box that prompts for a file.
- `openFile(filename)`: Opens the specified file.
- `openURL()`: Opens a dialog box that prompts for a URL.
- `openURL(URL)`: Opens the specified document. See EXAMPLES.
- `openURL(URL, new window)`: Opens a new window displaying the specified document.
- `saveAs()`: Opens a dialog box that prompts for a URL.
- `saveAs(output_file)`: Writes HTML to the specified file.
- `saveAs(output_file, type)`: Writes the type to the specified file (HTML, text, or PostScript).

**EXIT STATUS**

If a command fails, an error message is reported to stderr and the command exits with a nonzero status.

**EXAMPLES**

The following are all examples of using the `-remote` command option. For more information and examples, see http://home.netscape.com/newsref/std/x-remote.html

**EXAMPLE 1** Selecting among open Netscape windows

```bash
example$ netscape -id 0x3c00124 -remote 'openURL(http://www.sun.com)'
```
EXAMPLE 2 Adding a bookmark without raising a window

To add a bookmark without raising a window, followed by opening a URL and raising the window, enter:

```bash
example$ netscape -noraise -remote 'addBookmark(http://www.sun.com)'
-raise -remote 'openURL(http://home.netscape.com)'
```

EXAMPLE 3 Adding a specified document to the Bookmark list

```bash
example$ netscape -remote 'addBookmark(http://www.sun.com)'
```

EXAMPLE 4 Opening a specified document

```bash
example$ netscape -remote 'openURL(http://www.sun.com)'
```

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>NSCPcom</td>
</tr>
</tbody>
</table>

SEE ALSO

attributes(5)

Refer to the Netscape Communicator online help for more information.
newform(1)

NAME
newform – change the format of a text file

SYNOPSIS
[-cchar] [-1n] [filename...]

DESCRIPTION
newform reads lines from the named filenames, or the standard input if no input file is
named, and reproduces the lines on the standard output. Lines are reformatted in
accordance with command line options in effect.

Except for -s, command line options may appear in any order, may be repeated, and
may be intermingled with the optional filenames. Command line options are processed
in the order specified. This means that option sequences like "-e15 -160" will yield
results different from "-160 -e15". Options are applied to all filenames on the
command line.

OPTIONS
The following options are supported:

-s
Shears off leading characters on each line up to the first tab and
places up to 8 of the sheared characters at the end of the line. If
more than 8 characters (not counting the first tab) are sheared, the
eighth character is replaced by a * and any characters to the right
of it are discarded. The first tab is always discarded.

An error message and program exit will occur if this option is used
on a file without a tab on each line. The characters sheared off are
saved internally until all other options specified are applied to that
line. The characters are then added at the end of the processed
line.

For example, to convert a file with leading digits, one or more tabs,
and text on each line, to a file beginning with the text, all tabs after
the first expanded to spaces, padded with spaces out to column 72
(or truncated to column 72), and the leading digits placed starting
at column 73, the command would be:

newform -s -i -l -a -e filename

-itabspec
Input tab specification: expands tabs to spaces, according to the tab
specifications given. Tabspec recognizes all tab specification forms
described in tabs(1). In addition, tabspec may be -, in which
newform assumes that the tab specification is to be found in the
first line read from the standard input (see fspec(4)). If no tabspec
is given, tabspec defaults to -8. A tabspec of -0 expects no tabs; if
any are found, they are treated as -1.

-otabspec
Output tab specification: replaces spaces by tabs, according to the
tab specifications given. The tab specifications are the same as for
-itabspec. If no tabspec is given, tabspec defaults to -8. A tabspec of
-0 means that no spaces will be converted to tabs on output.
Truncate $n$ characters from the beginning of the line when the line length is greater than the effective line length (see `-ln`). Default is to truncate the number of characters necessary to obtain the effective line length. The default value is used when `-b` with no $n$ is used. This option can be used to delete the sequence numbers from a COBOL program as follows:

```
newform -l1 -b7 filename
```

Same as `-bn` except that characters are truncated from the end of the line.

Prefix $n$ characters (see `-cchar`) to the beginning of a line when the line length is less than the effective line length. Default is to prefix the number of characters necessary to obtain the effective line length.

Same as `-pn` except characters are appended to the end of a line.

Write the tab specification format line on the standard output before any other lines are output. The tab specification format line which is printed will correspond to the format specified in the last `-o` option. If no `-o` option is specified, the line which is printed will contain the default specification of `-8`.

Change the prefix/append character to `char`. Default character for `char` is a space.

Set the effective line length to $n$ characters. If $n$ is not entered, `-l` defaults to 72. The default line length without the `-l` option is 80 characters. Note: Tabs and backspaces are considered to be one character (use `-i` to expand tabs to spaces).

The `-ll` must be used to set the effective line length shorter than any existing line in the file so that the `-b` option is activated.

**OPERANDS**

The following operand is supported:

```
filename  Input file
```

**EXIT STATUS**

The following exit values are returned:

<table>
<thead>
<tr>
<th>Exit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful operation.</td>
</tr>
<tr>
<td>1</td>
<td>Operation failed.</td>
</tr>
</tbody>
</table>

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
</table>
SEE ALSO
csplit(1), tabs(1), fspec(4), attributes(5)

DIAGNOSTICS
All diagnostics are fatal.

usage: . . .
    newform was called with a bad option.

"not -s format"
    There was no tab on one line.

"can't open file"
    Self-explanatory.

"internal line too long"
    A line exceeds 512 characters after being expanded in the internal work buffer.

"tabspec in error"
    A tab specification is incorrectly formatted, or specified tab stops are not ascending.

"tabspec indirection illegal"
    A tabspec read from a file (or standard input) may not contain a tabspec referencing another file (or standard input).

NOTES
newform normally only keeps track of physical characters; however, for the -i and -o options, newform will keep track of backspaces in order to line up tabs in the appropriate logical columns.

newform will not prompt the user if a tabspec is to be read from the standard input (by use of -i- or -o-).

If the -f option is used, and the last -o option specified was -o-, and was preceded by either a -o- or a -i-, the tab specification format line will be incorrect.
newgrp – log in to a new group

Command
/usr/bin/newgrp [-l] [group]

sh Built-in  newgrp  [argument]
ksh Built-in  *newgrp  [argument]

The newgrp command logs a user into a new group by changing a user’s real and effective group ID. The user remains logged in and the current directory is unchanged. The execution of newgrp always replaces the current shell with a new shell, even if the command terminates with an error (unknown group).

Any variable that is not exported is reset to null or its default value. Exported variables retain their values. System variables (such as PS1, PS2, PATH, MAIL, and HOME), are reset to default values unless they have been exported by the system or the user. For example, when a user has a primary prompt string (PS1) other than $ (default) and has not exported PS1, the user’s PS1 will be set to the default prompt string $, even if newgrp terminates with an error. Note that the shell command export (see sh(1) and set(1)) is the method to export variables so that they retain their assigned value when invoking new shells.

With no operands and options, newgrp changes the user’s group IDs (real and effective) back to the group specified in the user’s password file entry. This is a way to exit the effect of an earlier newgrp command.

A password is demanded if the group has a password and the user is not listed in /etc/group as being a member of that group. The only way to create a password for a group is to use passwd(1), then cut and paste the password from /etc/shadow to /etc/group. Group passwords are antiquated and not often used.

sh Built-in  Equivalent to exec newgrp argument where argument represents the options and/or operand of the newgrp command.

ksh Built-in  Equivalent to exec to /bin/newgrp argument where argument represents the options and/or operand of the newgrp command.

On this man page, ksh(1) commands that are preceded by one or two * (asterisks) are treated specially in the following ways:

1. Variable assignment lists preceding the command remain in effect when the command completes.
2. I/O redirections are processed after variable assignments.
3. Errors cause a script that contains them to abort.
4. Words, following a command preceded by ** that are in the format of a variable assignment, are expanded with the same rules as a variable assignment. This means that tilde substitution is performed after the = sign and word splitting and file name generation are not performed.

OPTIONS
The following option is supported:
- Change the environment to what would be expected if the user actually logged in again as a member of the new group.

**OPERANDS**

The following operands are supported:

- **group**
  
  A group name from the group database or a non-negative numeric group ID. Specifies the group ID to which the real and effective group IDs will be set. If `group` is a non-negative numeric string and exists in the group database as a group name (see `getgrnam(3C)`), the numeric group ID associated with that group name will be used as the group ID.

- **argument**
  
  `sh` and `ksh` only. Options and/or operand of the `newgrp` command.

**ENVIRONMENT VARIABLES**

See `environ(5)` for descriptions of the following environment variables that affect the execution of `newgrp`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

**EXIT STATUS**

If `newgrp` succeeds in creating a new shell execution environment, whether or not the group identification was changed successfully, the exit status will be the exit status of the shell. Otherwise, the following exit value is returned:

- `>0`  
  An error occurred.

**FILES**

- `/etc/group`  
  system’s group file

- `/etc/passwd`  
  system’s password file

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>Attribute Type</th>
<th>Attribute Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

**SEE ALSO**

`login(1)`, `ksh(1)`, `set(1)`, `sh(1)`, `intc(3)`, `getgrnam(3C)`, `group(4)`, `passwd(4)`, `attributes(5)`, `environ(5)`
NAME
news – print news items

SYNOPSIS
news [-a] [-n] [-s] [items]

DESCRIPTION
news is used to keep the user informed of current events. By convention, these events
are described by files in the directory /var/news.

When invoked without arguments, news prints the contents of all current files in
/var/news, most recent first, with each preceded by an appropriate header. news
stores the “currency” time as the modification date of a file named .news_time in
the user’s home directory (the identity of this directory is determined by the
environment variable $HOME); only files more recent than this currency time are
considered “current.”

OPTIONS
-a  Print all items, regardless of currency. In this case, the stored time is not
changed.

-n  Report the names of the current items without printing their contents, and
without changing the stored time.

-s  report how many current items exist, without printing their names or
contents, and without changing the stored time. It is useful to include such
an invocation of news in one’s .profile file, or in the system’s
/etc/profile.

All other arguments are assumed to be specific news items that are to be printed.

If a delete is typed during the printing of a news item, printing stops and the next item
is started. Another delete within one second of the first causes the program to
terminate.

ENVIRONMENT
VARIABLES
See environ(5) for a description of the LC_CTYPE environment variable that affects
the execution of news.

FILES
/etc/profile
/var/news/*
$HOME/.news_time

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

SEE ALSO
profile(4), attributes(5), environ(5)
newtask(1)

NAME
newtask – create new task or change task or project of running process

SYNOPSIS
newtask [-c pid | [-F] [command...]] [-p project] [-v]

DESCRIPTION
The newtask command executes the user’s default shell or a specified command, placing the executed command in a new task owned by the specified project. The user’s default shell is the one specified in the passwd database, and is determined via getpwnam().

Alternatively, newtask may be used to change the task of an already running process. A project may also be specified in this form of the command. This may be desirable for processes that are mission critical and cannot be restarted in order to put them into a new project.

In the case that extended accounting is active, the newtask command may additionally cause the creation of a task accounting record marking the completion of the preceding system task.

OPTIONS
The following options are supported:

- \(c\) pid Changes the task or project of a running process. The invoking user must either own the process or have super-user privileges.

If the process is being changed, the process owner must be a member of the specified process. When the process is changed for a running process, its pool binding as well as resource controls are modified to match the configuration of the new project. Controls not explicitly specified in the project entry will be preserved.

This option is incompatible with the \(-F\) and \(-l\) options.

- \(F\) Creates a finalized task, within which further newtask or settaskid(2) invocations would fail. Finalized tasks may be useful at some sites for simplifying the attribution of resource consumption.

- \(l\) Changes the environment to what would be expected if the user actually logged in again as a member of the new project.

- \(p\) Changes the project ID of the new task to that associated with the given project name. The invoking user must be a valid member of the requested project for the command to succeed. If no project name is specified, the new task is started in the invoking user’s current project.

- \(v\) Verbose: displays the system task id as the new system task is begun.

OPERANDS
The following operands are supported:

project The project to which resource usage by the created task should be charged. The requested project must be defined in the project databases defined in nsswitch.conf(4).

command The command to be executed as the new task. If no command is given, the user’s login shell is invoked. (If the login shell is not
newtask(1)

available, /bin/sh is invoked.)

EXAMPLES

EXAMPLE 1 Creating a new shell

The following example creates a new shell in the canada project, displaying the task id:

```
example$ id -p
uid=565(gh) gid=10(staff) projid=10(default)
example$ newtask -v -p canada
38
example$ id -p
uid=565(gh) gid=10(staff) projid=82(canada)
```

EXAMPLE 2 Running the date command

The following example runs the date command in the russia project:

```
example$ newtask -p russia date
Tue Aug 31 11:12:10 PDT 1999
```

EXAMPLE 3 Changing the project of an existing process

The following example changes the project of the existing process with a pid of 9999 to russia:

```
example$ newtask -c 9999 -p russia
```

EXIT STATUS

The following exit values are returned:

0  Successful execution.
1  A fatal error occurred during execution.
2  Invalid command line options were specified.

FILES

/etc/project  Local database containing valid project definitions for this machine.
/proc/pid/*  Process information and control files.

ATTRIBUTES

See attributes(5) for a description of the following attributes:

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO

proc(1), id(1M), poolbind(1M), execvp(2), setrlimit(2), settaskid(2), setproject(3PROJECT), nsswitch.conf(4), proc(4), project(4), attributes(5)
nice(1)

NAME
nice – invoke a command with an altered scheduling priority

SYNOPSIS
/usr/bin/nice [-increment | -n increment] command [argument...]
/usr/xpg4/bin/nice [-increment | -n increment] command [argument...]
csh Builtin
nice [-increment | +increment] [command]

DESCRIPTION
The nice utility invokes command, requesting that it be run with a different system scheduling priority. The priocntl(1) command is a more general interface to scheduler functions.

The invoking process (generally the user’s shell) must be in a scheduling class that supports nice.

If the C shell (see csh(1)) is used, the full path of the command must be specified; otherwise, the csh built-in version of nice will be invoked. See csh Builtin below.

/usr/bin/nice
If nice executes commands with arguments, it uses the default shell /usr/bin/sh (see sh(1)).

/usr/xpg4/bin/nice
If nice executes commands with arguments, it uses /usr/xpg4/bin/sh (see ksh(1)).

csh Builtin
nice is also a csh built-in command with behavior different from the utility versions. See csh(1) for description.

OPTIONS
The following options are supported:

-increment | -n increment
increment must be in the range 1-19; if not specified, an increment of 10 is assumed. An increment greater than 19 is equivalent to 19.

The super-user may run commands with priority higher than normal by using a negative increment such as -10. A negative increment assigned by an unprivileged user is ignored.

OPERANDS
The following operands are supported:

command The name of a command that is to be invoked. If command names any of the special built-in utilities (see shell builtins(1)), the results are undefined.

argument Any string to be supplied as an argument when invoking command.

ENVIRONMENT VARIABLES
See environ(5) for descriptions of the following environment variables that affect the execution of nice: LC_CTYPE, LC_MESSAGES, PATH, and NLSPATH.

EXIT STATUS
If command is invoked, the exit status of nice will be the exit status of command; otherwise, nice will exit with one of the following values:

1-125 An error occurred.
126   *command* was found but could not be invoked.
127   *command* could not be found.

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>/usr/bin/nice</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTRIBUTES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/usr/bin/nice</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATTRIBUTES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/usr/xpg4/bin/nice</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATTRIBUTES</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SEE ALSO csh(1), ksh(1), nchup(1), priocntl(1), sh(1), shell_builtins(1), nice(2), attributes(5), environ(5), XPG4(5)
NIS+ is a new version of the network information nameservice. This version differs in several significant ways from version 2, which is referred to as NIS or YP in earlier releases. Specific areas of enhancement include the ability to scale to larger networks, security, and the administration of the service.

The man pages for NIS+ are broken up into three basic categories. Those in section 1 are the user commands that are most often executed from a shell script or directly from the command line. Section 1M man pages describe utility commands that can be used by the network administrator to administer the service itself. The NIS+ programming API is described by man pages in section 3NSL.

All commands and functions that use NIS version 2 are prefixed by the letters yp as in ypmatch(1), ypcat(1), yp_match(3NSL), and yp_first(3NSL). Commands and functions that use the new replacement software NIS+ are prefixed by the letters nis as in nismatch(1), nischown(1), nis_list(3NSL), and nis_add_entry(3NSL). A complete list of NIS+ commands is in the LIST OF COMMANDS section.

This man page introduces the NIS+ terminology. It also describes the NIS+ namespace, authentication, and authorization policies.

The naming model of NIS+ is based upon a tree structure. Each node in the tree corresponds to an NIS+ object. There are six types of NIS+ objects: directory, table, group, link, entry, and private.

Each NIS+ namespace will have at least one NIS+ directory object. An NIS+ directory is like a UNIX file system directory which contains other NIS+ objects including NIS+ directories. The NIS+ directory that forms the root of the NIS+ namespace is called the root directory. There are two special NIS+ directories: org_dir and groups_dir. The org_dir directory consists of all the system-wide administration tables, such as passwd, hosts, and mail_aliases. The groups_dir directory consists of NIS+ group objects which are used for access control. The collection of org_dir, groups_dir and their parent directory is referred to as an NIS+ domain. NIS+ directories can be arranged in a tree-like structure so that the NIS+ namespace can match the organizational or administrative hierarchy.

NIS+ tables (not files), contained within NIS+ directories, store the actual information about some particular type. For example, the hosts system table stores information about the IP address of the hosts in that domain. NIS+ tables are multicolumn and the tables can be searched through any of the searchable columns. Each table object defines the schema for its table. The NIS+ tables consist of NIS+ entry objects. For each entry in the NIS+ table, there is an NIS+ entry object. NIS+ entry objects conform to the schema defined by the NIS+ table object.

NIS+ group objects are used for access control at group granularity. NIS+ group objects, contained within the groups_dir directory of a domain, contain a list of all the NIS+ principals within a certain NIS+ group. An NIS+ principal is a user or a machine making NIS+ requests.
**NIS+ Link Object**

NIS+ link objects are like UNIX symbolic file-system links—they are typically used for shortcuts in the NIS+ namespace. Refer to `nis_objects(3NSL)` for more information about the NIS+ objects.

**NIS+ NAMES**

The NIS+ service defines two forms of names, simple names and indexed names. Simple names are used by the service to identify NIS+ objects contained within the NIS+ namespace. Indexed names are used to identify NIS+ entries contained within NIS+ tables. Furthermore, entries within NIS+ tables are returned to the caller as NIS+ objects of type `entry`. NIS+ objects are implemented as a union structure which is described in the file `<rpcsvc/nis_object.x>`. The differences between the various types and the meanings of the components of these objects are described in `nis_objects(3NSL)`.

**Simple Names**

Simple names consist of a series of labels that are separated by the `. (dot)` character. Each label is composed of printable characters from the ISO Latin 1 set. Each label can be of any nonzero length, provided that the fully qualified name is fewer than `NIS_MAXNAMELEN` octets including the separating dots. (See `<rpcsvc/nis.h>` for the actual value of `NIS_MAXNAMELEN` in the current release.) Labels that contain special characters (see Grammar) must be quoted.

The NIS+ namespace is organized as a singly rooted tree. Simple names identify nodes within this tree. These names are constructed such that the leftmost label in a name identifies the leaf node and all of the labels to the right of the leaf identify that object’s parent node. The parent node is referred to as the leaf’s directory. This is a naming directory and should not be confused with a file system directory.

For example, the name `example.simple.name` is a simple name with three labels, where `example` is the leaf node in this name, the directory of this leaf is `simple.name`, which by itself is a simple name. The leaf of which is `simple` and its directory is simply `name`.

The function `nis_leaf_of(3NSL)` returns the first label of a simple name. The function `nis_domain_of(3NSL)` returns the name of the directory that contains the leaf. Iterative use of these two functions can break a simple name into each of its label components.

The name `. (dot)` is reserved to name the global root of the namespace. For systems that are connected to the Internet, this global root will be served by a Domain Name Service. When an NIS+ server is serving a root directory whose name is not `. (dot)` this directory is referred to as a local root.

NIS+ names are said to be fully qualified when the name includes all of the labels identifying all of the directories, up to the global root. Names without the trailing dot are called partially qualified.

**Indexed Names**

Indexed names are compound names that are composed of a search criterion and a simple name. The search criterion component is used to select entries from a table; the simple name component is used to identify the NIS+ table that is to be searched. The search criterion is a series of column names and their desired values enclosed in bracket `['']` characters. These criteria take the following form:
A search criterion is combined with a simple name to form an indexed name by concatenating the two parts, separated by a ‘,’ (comma) character as follows.

```
[ search-criterion ], table.directory
```

When multiple column name/value pairs are present in the search criterion, only those entries in the table that have the appropriate value in all columns specified are returned. When no column name/value pairs are specified in the search criterion, [], all entries in the table are returned.

**Grammar**

The following text represents a context-free grammar that defines the set of legal NIS+ names. The terminals in this grammar are the characters ‘.’ (dot), ‘[’ (open bracket), ‘]’ (close bracket), ‘,’ (comma), ‘=’ (equals) and whitespace. Angle brackets (< and >), which delineate non-terminals, are not part of the grammar. The character ‘|’ (vertical bar) is used to separate alternate productions and should be read as "this production OR this production".

```
name ::= . | <simple name> | <indexed name>
simple name ::= <string>. | <string>, <simple name>
indexed name ::= <search criterion>, <simple name>
search criterion ::= [ <attribute list> ]
attribute list ::= <attribute> | <attribute>, <attribute list>
attribute ::= <string> = <string>
string ::= ISO Latin 1 character set except the character ‘/’ (slash). The initial character may not be a terminal character or the characters ‘@’ (at), ‘+’ (plus), or ‘-’ (hyphen).
```

Terminals that appear in strings must be quoted with "" (double quote). The ‘"’ character may be quoted by quoting it with itself "".

**Name Expansion**

The NIS+ service only accepts fully qualified names. However, since such names may be unwieldy, the NIS+ commands in section 1 employ a set of standard expansion rules that will attempt to fully qualify a partially qualified name. This expansion is actually done by the NIS+ library function nis_getnames(3NSL) which generates a list of names using the default NIS+ directory search path or the NIS_PATH environment variable. The default NIS+ directory search path includes all the names in its path. nis_getnames() is invoked by the functions nis_lookup(3NSL) and nis_list(3NSL) when the EXPAND_NAME flag is used.

nis+(1)
The **NIS_PATH** environment variable contains an ordered list of simple names. The names are separated by the ‘:’ (colon) character. If any name in the list contains colons, the colon should be quoted as described in the Grammar section. When the list is exhausted, the resolution function returns the error **NIS_NOTFOUND**. This may mask the fact that the name existed but a server for it was unreachable. If the name presented to the list or lookup interface is fully qualified, the **EXPAND_NAME** flag is ignored.

In the list of names from the **NIS_PATH** environment variable, the ‘$’ (dollar sign) character is treated specially. Simple names that end with the label ‘$’ have this character replaced by the default directory (see **nis_local_directory**(3NSL)). Using "$" as a name in this list results in this name being replaced by the list of directories between the default directory and the global root that contain at least two labels.

Below is an example of this expansion. Given the default directory of `some.long.domain.name`, and the **NIS_PATH** variable set to `fred.bar:org_dir.$::`. This path is initially broken up into the list:

```
1   fred.bar.
2   org_dir.$
3   $
```

The dollar sign in the second component is replaced by the default directory. The dollar sign in the third component is replaced with the names of the directories between the default directory and the global root that have at least two labels in them. The effective path value becomes:

```
1   fred.bar.
2a  org_dir.some.long.domain.name.
3a  some.long.domain.name.
3b  long.domain.name.
3c  domain.name.
```

Each of these simple names is appended to the partially qualified name that was passed to the **nis_lookup**(3NSL) or **nis_list**(3NSL) interface. Each is tried in turn until **NIS_SUCCESS** is returned or the list is exhausted.

If the **NIS_PATH** variable is not set, the path ‘$’ is used.

The library function **nis_getnames**(3NSL) can be called from user programs to generate the list of names that would be attempted. The program **nisdefaults**(1) with the `-s` option can also be used to show the fully expanded path.

Normally, all the entries for a certain type of information are stored within the table itself. However, there are times when it is desirable for the table to point to other tables where entries can be found. For example, you may want to store all the IP
addresses in the host table for their own domain, and yet want to be able to resolve hosts in some other domain without explicitly specifying the new domain name. NIS+ provides a mechanism for concatenating different but related tables with a "NIS+ Concatenation Path". With a concatenation path, you can create a sort of flat namespace from a hierarchical structure. You can also create a table with no entries and just point the hosts or any other table to its parent domain. Note that with such a setup, you are moving the administrative burden of managing the tables to the parent domain. The concatenation path will slow down the request response time because more tables and more servers are searched. It will also decrease the availability if all the servers are incapacitated for a particular directory in the table path.

The NIS+ Concatenation Path is also referred to as the "table path". This path is set up at table creation time through nistbladm(1). You can specify more than one table to be concatenated and they will be searched in the given order. Note that the NIS+ client libraries, by default, will not follow the concatenation path set in site-specific tables. Refer to nis_list(3NSL) for more details.

**Namespaces**

The NIS+ service defines two additional disjoint namespaces for its own use. These namespaces are the NIS+ Principal namespace, and the NIS+ Group namespace. The names associated with the group and principal namespaces are syntactically identical to simple names. However, the information they represent cannot be obtained by directly presenting these names to the NIS+ interfaces. Instead, special interfaces are defined to map these names into NIS+ names so that they may then be resolved.

**Principal Names**

NIS+ principal names are used to uniquely identify users and machines that are making NIS+ requests. These names have the form:

```
principal.domain
```

Here `domain` is the fully qualified name of an NIS+ directory where the named principal’s credentials can be found. See Directories and Domains for more information on domains. Note that in this name, `principal`, is not a leaf in the NIS+ namespace.

Credentials are used to map the identity of a host or user from one context such as a process UID into the NIS+ context. They are stored as records in an NIS+ table named `cred`, which always appears in the `org_dir` subdirectory of the directory named in the principal name.

This mapping can be expressed as a replacement function:

```
principal.domain -> [cname=principal.domain], cred.org_dir.domain
```

This latter name is an NIS+ name that can be presented to the nis_list(3NSL) interface for resolution. NIS+ principal names are administered using the nisaddcred(1M) command.
The cred table contains five columns named `cname`, `auth_name`, `auth_type`, `public_data`, and `private_data`. There is one record in this table for each identity mapping for an NIS+ principal. The current service supports three types of mappings:

**LOCAL**
This mapping is used to map from the UID of a given process to the NIS+ principal name associated with that UID. If no mapping exists, the name nobody is returned. When the effective UID of the process is 0 (for example, the superuser), the NIS+ name associated with the host is returned. Note that UIDs are sensitive to the context of the machine on which the process is executing.

**DES**
This mapping is used to map to and from a Secure RPC "netname" into an NIS+ principal name. See secure_rpc(3NSL) for more information on netnames. Note that since netnames contain the notion of a domain, they span NIS+ directories.

**DHnnn-m**
Example: DH640-0, DH1024-0. Analogous to DES mappings, these are used to map netnames and NIS+ principal names for extended Diffie-Hellman keys. See nisauthconf(1M) for further information.

The NIS+ client library function `nis_local_principal(3NSL)` uses the `cred.org_dir` table to map the UNIX notion of an identity, a process' UID, into an NIS+ principal name. Shell programs can use the program nisdefaults(1) with the -p switch to return this information.

Mapping from UIDs to an NIS+ principal name is accomplished by constructing a query of the form:

```
[auth_type=LOCAL, auth_name=uid],cred.org_dir.default-domain.
```

This query will return a record containing the NIS+ principal name associated with this UID, in the machine's default domain.

The NIS+ service uses the DES mapping to map the names associated with Secure RPC requests into NIS+ principal names. RPC requests that use Secure RPC include the netname of the client making the request in the RPC header. This netname has the form:

```
unix.UID@domain
```

The service constructs a query using this name of the form:

```
[auth_type=DES, auth_name=netname],cred.org_dir.domain.
```

where the domain part is extracted from the netname rather than using the default domain. This query is used to look up the mapping of this netname into an NIS+ principal name in the domain where it was created.
This mechanism of mapping UID and netnames into an NIS+ principal name guarantees that a client of the NIS+ service has only one principal name. This principal name is used as the basis for authorization which is described below. All objects in the NIS+ namespace and all entries in NIS+ tables must have an owner specified for them. This owner field always contains an NIS+ principal name.

**Group Names**

Like NIS+ principal names, NIS+ group names take the form:

```
group_name.domain
```

All objects in the NIS+ namespace and all entries in NIS+ tables may optionally have a group owner specified for them. This group owner field, when filled in, always contains the fully qualified NIS+ group name.

The NIS+ client library defines several interfaces (nis_groups(3NSL)) for dealing with NIS+ groups. These interfaces internally map NIS+ group names into an NIS+ simple name which identifies the NIS+ group object associated with that group name. This mapping can be shown as follows:

```
group.domain -> group.groups_dir.domain
```

This mapping eliminates collisions between NIS+ group names and NIS+ directory names. For example, without this mapping, a directory with the name `engineering.foo.com` would make it impossible to have a group named `engineering.foo.com`. This is due to the restriction that within the NIS+ namespace, a name unambiguously identifies a single object. With this mapping, the NIS+ group name `engineering.foo.com` maps to the NIS+ object name `engineering.groups_dir.foo.com`.

The contents of a group object is a list of NIS+ principal names, and the names of other NIS+ groups. See nis_groups(3NSL) for a more complete description of their use.

**NIS+ SECURITY**

NIS+ defines a security model to control access to information managed by the service. The service defines access rights that are selectively granted to individual clients or groups of clients. Principal names and group names are used to define clients and groups of clients that may be granted or denied access to NIS+ information. These principals and groups are associated with NIS+ domains as defined below.

The security model also uses the notion of a class of principals called `nobody`, which contains all clients, whether or not they have authenticated themselves to the service. The class `world` includes any client who has been authenticated.

**Directories and Domains**

Some directories within the NIS+ namespace are referred to as NIS+ Domains. Domains are those NIS+ directories that contain the subdirectories `groups_dir` and `org_dir`. Further, the subdirectory `org_dir` should contain the table named `cred`. NIS+ Group names and NIS+ Principal names always include the NIS+ domain name after their first label.
The NIS+ name service uses Secure RPC for the integrity of the NIS+ service. This requires that users of the service and their machines must have a Secure RPC key pair associated with them. This key is initially generated with either the nisaddcred(1M) or nisclient(1M) commands and modified with the chkey(1) or nispasswd(1) commands.

The use of Secure RPC allows private information to be stored in the name service that will not be available to untrusted machines or users on the network.

In addition to the Secure RPC key, users need a mapping of their UID into an NIS+ principal name. This mapping is created by the system administrator using either the nisclient(1M) or the nisaddcred(1M) command.

Users that will be using machines in several NIS+ domains must insure that they have a local credential entry in each of those domains. This credential should be created with the NIS+ principal name of the user in the user’s “home” domain. For the purposes of NIS+ and Secure RPC, the home domain is defined to be the one where the user's Secure RPC key pair is located.

Although extended Diffie-Hellman keys use an alternative to Secure RPC, administration is done through the same commands. See nisauthconf(1M).

The NIS+ service defines four access rights that can be granted or denied to clients of the service. These rights are read, modify, create, and destroy. These rights are specified in the object structure at creation time and may be modified later with the nischmod(1) command. In general, the rights granted for an object apply only to that object. However, for purposes of authorization, rights granted to clients reading directory and table objects are granted to those clients for all of the objects “contained” by the parent object. This notion of containment is abstract. The objects do not actually contain other objects within them. Note that group objects do contain the list of principals within their definition.

Access rights are interpreted as follows:

<table>
<thead>
<tr>
<th>Access Right</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>read</td>
<td>This right grants read access to an object. For directory and table objects, having read access on the parent object conveys read access to all of the objects that are direct children of a directory, or entries within a table.</td>
</tr>
<tr>
<td>modify</td>
<td>This right grants modification access to an existing object. Read access is not required for modification. However, in many applications, one will need to read an object before modifying it. Such modify operations will fail unless read access is also granted.</td>
</tr>
<tr>
<td>create</td>
<td>This right gives a client permission to create new objects where one had not previously existed. It is only used in conjunction with directory and table objects. Having create access for a table allows a client to add additional entries to the table. Having create access for a directory allows a client to add new objects to an NIS+ directory.</td>
</tr>
</tbody>
</table>
destroy  This right gives a client permission to destroy or remove an
existing object or entry. When a client attempts to destroy an entry
or object by removing it, the service first checks to see if the table
or directory containing that object grants the client destroy access.
If it does, the operation proceeds, if the containing object does not
grant this right then the object itself is checked to see if it grants
this right to the client. If the object grants the right, then the
operation proceeds; otherwise the request is rejected.

Each of these rights may be granted to any one of four different categories.

owner  A right may be granted to the owner of an object. The owner is the
NIS+ principal identified in the owner field. The owner can be
changed with the nischown(1) command. Note that if the owner
does not have modification access rights to the object, the owner
cannot change any access rights to the object, unless the owner has
modification access rights to its parent object.

group owner  A right may be granted to the group owner of an object. This grants
the right to any principal that is identified as a member of the
group associated with the object. The group owner may be
changed with the nischgrp(1) command. The object owner need
not be a member of this group.

world  A right may be granted to everyone in the world. This grants the
right to all clients who have authenticated themselves with the
service.

nobody  A right may be granted to the nobody principal. This has the effect
of granting the right to any client that makes a request of the
service, regardless of whether they are authenticated or not.

Note that for bootstrapping reasons, directory objects that are NIS+ domains, the
org_dir subdirectory and the cred table within that subdirectory must have read access
to the nobody principal. This makes navigation of the namespace possible when a client
is in the process of locating its credentials. Granting this access does not allow the
contents of other tables within org_dir to be read (such as the entries in the password
table) unless the table itself gives "real" access rights to the nobody principal.

Additional capabilities are provided for granting access rights to clients for directories.
These rights are contained within the object access rights (OAR) structure of the
directory. This structure allows the NIS+ service to grant rights that are not granted by
the directory object to be granted for objects contained by the directory of a specific
type.

An example of this capability is a directory object which does not grant create access
to all clients, but does grant create access in the OAR structure for group type objects to
clients who are members of the NIS+ group associated with the directory. In this
example the only objects that could be created as children of the directory would have
to be of the type group.
Another example is a directory object that grants create access only to the owner of the directory, and then additionally grants create access through the OAR structure for objects of type table, link, group, and private to any member of the directory’s group. This has the effect of giving nearly complete create access to the group with the exception of creating subdirectories. This restricts the creation of new NIS+ domains because creating a domain requires creating both a groups_dir and org_dir subdirectory.

Note that there is currently no command line interface to set or change the OAR of the directory object.

As with directories, additional capabilities are provided for granting access to entries within tables. Rights granted to a client by the access rights field in a table object apply to the table object and all of the entry objects “contained” by that table. If an access right is not granted by the table object, it may be granted by an entry within the table. This holds for all rights except create.

For example, a table may not grant read access to a client performing a nis_list(3NSL) operation on the table. However, the access rights field of entries within that table may grant read access to the client. Note that access rights in an entry are granted to the owner and group owner of the entry and not the owner or group of the table. When the list operation is performed, all entries that the client has read access to are returned. Those entries that do not grant read access are not returned. If none of the entries that match the search criterion grant read access to the client making the request, no entries are returned and the result status contains the NIS_NOTFOUND error code.

Access rights that are granted by the rights field in an entry are granted for the entire entry. However, in the table object an additional set of access rights is maintained for each column in the table. These rights apply to the equivalent column in the entry. The rights are used to grant access when neither the table nor the entry itself grant access. The access rights in a column specification apply to the owner and group owner of the entry rather than the owner and group owner of the table object.

When a read operation is performed, if read access is not granted by the table and is not granted by the entry but is granted by the access rights in a column, that entry is returned with the correct values in all columns that are readable and the string *NP* (No Permission) in columns where read access is not granted.

As an example, consider a client that has performed a list operation on a table that does not grant read access to that client. Each entry object that satisfied the search criterion specified by the client is examined to see if it grants read access to the client. If it does, it is included in the returned result. If it does not, then each column is checked to see if it grants read access to the client. If any columns grant read access to the client, data in those columns is returned. Columns that do not grant read access have their contents replaced by the string *NP*. If none of the columns grant read access, then the entry is not returned.
Most NIS+ operations have implied access control through the permissions on the objects that they manipulate. For example, in order to read an entry in a table, you must have read permission on that entry. However, some NIS+ operations by default perform no access checking at all and so are allowed for anyone.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Example of commands that use the operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIS_CHECKPOINT</td>
<td>nisping -C</td>
</tr>
<tr>
<td>NIS_CPTIME</td>
<td>nisping, rpc.nisd</td>
</tr>
<tr>
<td>NIS_MKDIR</td>
<td>nismkdir</td>
</tr>
<tr>
<td>NIS_PING</td>
<td>nisping, rpc.nisd</td>
</tr>
<tr>
<td>NIS_RMDIR</td>
<td>nisrmdir</td>
</tr>
<tr>
<td>NIS_SERVSTATE</td>
<td>nisbackup, nisrestore</td>
</tr>
<tr>
<td>NIS_STATUS</td>
<td>nisstat, rpc.nispasswd</td>
</tr>
</tbody>
</table>

See `nisopaccess(1)` for a description of how to enforce access control to these NIS+ operations.

The following lists all commands and programming functions related to NIS+:

- `nisaddent(1M)` add /etc files and NIS maps into their corresponding NIS+ tables
- `niscat(1)` display NIS+ tables and objects
- `nischgrp(1)` change the group owner of a NIS+ object
- `nischmod(1)` change access rights on a NIS+ object
- `nischown(1)` change the owner of a NIS+ object
- `nischttl(1)` change the time to live value of a NIS+ object
- `nisdefaults(1)` display NIS+ default values
- `niserror(1)` display NIS+ error messages
- `nissgrep(1)` utilities for searching NIS+ tables
- `nissgrpadm(1)` NIS+ group administration command
- `nisln(1)` symbolically link NIS+ objects
- `nisl(1)` list the contents of a NIS+ directory
- `nismatch(1)` utilities for searching NIS+ tables
- `nismkdir(1)` create NIS+ directories
- `nisopaccess(1)` access control for protocol operations
- `nispasswd(1)` change NIS+ password information
nisrm(1)  remove NIS+ objects from the namespace
nisrmkdir(1)  remove NIS+ directories
nisshowcache(1M)  NIS+ utility to print out the contents of the shared cache file
nistbladm(1)  NIS+ table administration command
nissetest(1)  return the state of the NIS+ namespace using a conditional expression
aliasadm(1M)  manipulate the NIS+ aliases map
nis_cachemgr(1M)  NIS+ utility to cache location information about NIS+ servers
nisaddcred(1M)  create NIS+ credentials
nisaddent(1M)  create NIS+ tables from corresponding /etc files or NIS+ maps
nisauthconf(1M)  configure extended Diffie-Hellman keys
nisbackup(1M)  backup NIS+ directories
nisclient(1M)  initialize NIS+ credentials for NIS+ principals
nisd(1M)  NIS+ service daemon
nisd_resolv(1M)  NIS+ service daemon
nisinit(1M)  NIS+ client and server initialization utility
nislog(1M)  display the contents of the NIS+ transaction log
nisping(1M)  send ping to NIS+ servers
nispopulate(1M)  populate the NIS+ tables in a NIS+ domain
nisprefadm(1M)  NIS+ utility to set server preferences for NIS+ clients
nisrestore(1M)  restore NIS+ directory backup
nisserver(1M)  set up NIS+ servers
nissetup(1M)  initialize a NIS+ domain
nisshowcache(1M)  NIS+ utility to print out the contents of the shared cache file
nisstat(1M)  report NIS+ server statistics
nisupdkeys(1M)  update the public keys in a NIS+ directory object
nis+(1)

NIS+ Programming API

- rpc.nisd(1M) NIS+ service daemon
- rpc.nisd_resolv(1M) NIS+ service daemon
- sysidns(1M) system configuration
- nis_add(3NSL) NIS+ namespace functions
- nis_add_entry(3NSL) NIS+ table functions
- nis_addmember(3NSL) NIS+ group manipulation functions
- nis_checkpoint(3NSL) miscellaneous NIS+ log administration functions
- nis_clone_object(3NSL) NIS+ subroutines
- nis_creategroup(3NSL) NIS+ group manipulation functions
- nis_destroy_object(3NSL) NIS+ subroutines
- nis_destroygroup(3NSL) NIS+ group manipulation functions
- nis_dir_cmp(3NSL) NIS+ subroutines
- nis_domain_of(3NSL) NIS+ subroutines
- nis_error(3NSL) display NIS+ error messages
- nis_first_entry(3NSL) NIS+ table functions
- nis_frenames(3NSL) NIS+ subroutines
- nis_freeresult(3NSL) NIS+ namespace functions
- nis_freeservlist(3NSL) miscellaneous NIS+ functions
- nis_freetags(3NSL) miscellaneous NIS+ functions
- nis_getnames(3NSL) NIS+ subroutines
- nis_getservlist(3NSL) miscellaneous NIS+ functions
- nis_groups(3NSL) NIS+ group manipulation functions
- nis_ismember(3NSL) NIS+ group manipulation functions
- nis_leaf_of(3NSL) NIS+ subroutines
- nis_lerror(3NSL) display some NIS+ error messages
- nis_list(3NSL) NIS+ table functions
- nis_local_directory(3NSL) NIS+ local names
- nis_local_group(3NSL) NIS+ local names
- nis_local_host(3NSL) NIS+ local names
- nis_local_names(3NSL) NIS+ local names
nis_local_principal(3NSL) NIS+ local names
nis_lookup(3NSL) NIS+ namespace functions
nis_mkdir(3NSL) miscellaneous NIS+ functions
nis_modify(3NSL) NIS+ namespace functions
nis_modify_entry(3NSL) NIS+ table functions
nis_name_of(3NSL) NIS+ subroutines
nis_names(3NSL) NIS+ namespace functions
nis_next_entry(3NSL) NIS+ table functions
nis_objects(3NSL) NIS+ object formats
nis_perror(3NSL) display NIS+ error messages
nis_ping(3NSL) miscellaneous NIS+ log administration functions
nis_print_group_entry(3NSL) NIS+ group manipulation functions
nis_print_object(3NSL) NIS+ subroutines
nis_remove(3NSL) NIS+ namespace functions
nis_remove_entry(3NSL) NIS+ table functions
nis_removemember(3NSL) NIS+ group manipulation functions
nis_rmdir(3NSL) miscellaneous NIS+ functions
nis_server(3NSL) miscellaneous NIS+ functions
nis_servstate(3NSL) miscellaneous NIS+ functions
display NIS+ error messages
nis_sperrno(3NSL) display NIS+ error messages
nis_sperror(3NSL) display NIS+ error messages
nis_sperror_r(3NSL) miscellaneous NIS+ functions
nis_stats(3NSL) NIS+ subroutines
nis_subr(3NSL) NIS+ table functions
nis_tables(3NSL) NIS+ group manipulation functions
nis_verifygroup(3NSL) NIS+ database files and directory structure
nisfiles(4) protocol description of an NIS+ object
deﬁnes the NIS+ protocol using the RPC
language as described in the ONC+
Developer’s Guide
nis+(1)

should be included by all clients of the NIS+ service

SEE ALSO
nischown(1), nisdefaults(1), nismatch(1), nisopaccess(1), nispasswd(1), newkey(1M), nisaddcred(1M), nisauthconf(1M), nisclient(1M), nispopulate(1M), nisserver(1M), nis_add_entry(3NSL), nis_domain_of(3NSL), nis_getnames(3NSL), nis_groups(3NSL), nis_leaf_of(3NSL), nis_list(3NSL), nis_local_directory(3NSL), nis_lookup(3NSL), nis_objects(3NSL)

System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)
- Describes how to make the transition from NIS to NIS+.

ONC+ Developer's Guide
- Describes the application programming interfaces for networks including NIS+.

System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)
- Describes how to plan for and configure an NIS+ namespace.

System Administration Guide, Volume 3
- Describes IPv6 extensions to Solaris name services.

NOTES
NIS+ might not be supported in future releases of the Solaris™ Operating Environment. Tools to aid the migration from NIS+ to LDAP are available in the Solaris 9 operating environment. For more information, visit http://www.sun.com/directory/nisplus/transition.html.
niscat – display NIS+ tables and objects

**SYNOPSIS**

```
niscat [-AhLMV] [-s sep] tablename...
niscat [-ALMP] -o name...
```

**DESCRIPTION**

In the first synopsis, `niscat` displays the contents of the NIS+ tables named by `tablename`. In the second synopsis, it displays the internal representation of the NIS+ objects named by `name`.

Columns without values in the table are displayed by two adjacent separator characters.

**OPTIONS**

The following options are supported:

- **-A** Display the data within the table and all of the data in tables in the initial table’s concatenation path.
- **-h** Display the header line prior to displaying the table. The header consists of the ‘#’ (hash) character followed by the name of each column. The column names are separated by the table separator character.
- **-L** Follow links. When this option is specified, if `tablename` or `name` names a LINK type object, the link is followed and the object or table named by the link is displayed.
- **-M** Master server only. This option specifies that the request should be sent to the master server of the named data. This guarantees that the most up-to-date information is seen at the possible expense of increasing the load on the master server and increasing the possibility of the NIS+ server being unavailable or busy for updates.
- **-P** Follow concatenation path. This option specifies that the request should follow the concatenation path of a table if the initial search is unsuccessful. This option is only useful when using an indexed name for `name` and the `-o` option.
- **-v** Display binary data directly. This option displays columns containing binary data on the standard output. Without this option binary data is displayed as the string `*BINARY*`.
- **-o name** Display the internal representation of the named NIS+ object(s). If `name` is an indexed name (see `nismatch(1)`), then each of the matching entry objects is displayed. This option is used to display access rights and other attributes of individual columns.
- **-s sep** This option specifies the character to use to separate the table columns. If no character is specified, the default separator for the table is used.
EXAMPLE 1 Using the niscat Command

This example displays the contents of the hosts table.

```
example% niscat -h hosts.org_dir
# cnname name addr comment
client1 client1 129.144.201.100 Joe Smith
crunchy crunchy 129.144.201.44 Jane Smith
```

The string *NP* is returned in those fields where the user has insufficient access rights.

Display the passwd.org_dir on the standard output.

```
example% niscat passwd.org_dir
```

Display the contents of table frodo and the contents of all tables in its concatenation path.

```
example% niscat -A frodo
```

Display the entries in the table groups.org_dir as NIS+ objects. Note that the brackets are protected from the shell by single quotes.

```
example% niscat -o '[ ]groups.org_dir'
```

Display the table object of the passwd.org_dir table.

```
example% niscat -o passwd.org_dir
```

The previous example displays the passwd table object and not the passwd table. The table object include information such as the number of columns, column type, searchable or not searchable separator, access rights, and other defaults.

Display the directory object for org_dir, which includes information such as the access rights and replica information.

```
example% niscat -o org_dir
```

ENVIRONMENT VARIABLES

NIS_PATH If this variable is set, and the NIS+ table name is not fully qualified, each directory specified will be searched until the table is found (see nisdefaults(1)).

EXIT STATUS

niscat returns the following values:

- 0 Successful completion
- 1 An error occurred.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:
NIS+ might not be supported in future releases of the Solaris™ Operating Environment. Tools to aid the migration from NIS+ to LDAP are available in the Solaris 9 operating environment. For more information, visit http://www.sun.com/directory/nisplus/transition.html.
nischgrp(1)

NAME     nischgrp – change the group owner of a NIS+ object

SYNOPSIS  nischgrp [-AfLP] group name...

DESCRIPTION nischgrp changes the group owner of the NIS+ objects or entries specified by name to the specified NIS+ group. Entries are specified using indexed names (see nismatch(1)). If group is not a fully qualified NIS+ group name, it will be resolved using the directory search path (see nisdefaults(1)).

The only restriction on changing an object's group owner is that you must have modify permissions for the object.

This command will fail if the master NIS+ server is not running.

The NIS+ server will check the validity of the group name prior to effecting the modification.

OPTIONS The following options are supported:

- A  Modify all entries in all tables in the concatenation path that match the search criterion specified in name. This option implies the - P switch.
- f  Force the operation and fail silently if it does not succeed.
- L  Follow links and change the group owner of the linked object or entries rather than the group owner of the link itself.
- P  Follow the concatenation path within a named table. This option only makes sense when either name is an indexed name or the - L switch is also specified and the named object is a link pointing to entries.

EXAMPLES  EXAMPLE 1 Using the nischgrp Command

The following two examples show how to change the group owner of an object to a group in a different domain, and how to change it to a group in the local domain, respectively.

example% nischgrp newgroup.remote.domain. object
example% nischgrp my-buds object

This example shows how to change the group owner for a password entry.

example% nischgrp admins ‘[uid=99],passwd.org_dir’

In the previous example, admins is a NIS+ group in the same domain.

The next two examples change the group owner of the object or entries pointed to by a link, and the group owner of all entries in the hobbies table.

example% nischgrp -L my-buds linkname
example% nischgrp my-buds ‘[]’,hobbies’
If this variable is set, and the NIS+ name is not fully qualified, each directory specified will be searched until the object is found (see nisdefaults(1)).

The following exit values are returned:

0 Successful operation.
1 Operation failed.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWnisu</td>
</tr>
</tbody>
</table>

NIS+ might not be supported in future releases of the Solaris™ Operating Environment. Tools to aid the migration from NIS+ to LDAP are available in the Solaris 9 operating environment. For more information, visit http://www.sun.com/directory/nisplus/transition.html.
nischmod(1)

NAME
nischmod – change access rights on a NIS+ object

SYNOPSIS
nischmod [-AFLP] mode name...

DESCRIPTION
nischmod changes the access rights (mode) of the NIS+ objects or entries specified by
name to mode. Entries are specified using indexed names (see nismatch(1)). Only
principals with modify access to an object may change its mode.

mode has the following form:

rights [, rights ] ...

rights has the form:

[ who ] op permission [ op permission ] ...

who is a combination of:

n    Nobody's permissions.

o    Owner's permissions.

g    Group's permissions.

w    World's permissions.

a    All, or owg.

If who is omitted, the default is a.

op is one of:

+    To grant the permission.

−    To revoke the permission.

=    To set the permissions explicitly.

permission is any combination of:

r    Read.

m    Modify.

c    Create.

d    Destroy.

Unlike the system chmod(1) command, this command does not accept an octal
notation.

OPTIONS
The following options are supported:

-A    Modify all entries in all tables in the concatenation path that match the
search criteria specified in name. This option implies the -P switch.
-f  Force the operation and fail silently if it does not succeed.
-L  Follow links and change the permission of the linked object or entries rather than the permission of the link itself.
-P  Follow the concatenation path within a named table. This option is only applicable when either name is an indexed name or the -L switch is also specified and the named object is a link pointing to an entry.

**EXAMPLES**

**EXAMPLE 1 Using the nischmod Command**

This example gives everyone read access to an object. (that is, access for owner, group, and all).

example$ nischmod a+r object

This example denies create and modify privileges to group and unauthenticated clients (nobody).

example$ nischmod gn-cm object

In this example, a complex set of permissions are set for an object.

example$ nischmod o=rmcd,g=rm,w=rc,n=r object

This example sets the permissions of an entry in the password table so that the group owner can modify them.

example$ nischmod g+m '[uid=55],passwd.org_dir'

The next example changes the permissions of a linked object.

example$ nischmod -L w+mr linkname

**ENVIRONMENT VARIABLES**

NIS_PATH  If this variable is set, and the NIS+ name is not fully qualified, each directory specified will be searched until the object is found (see nisdefaults(1)).

**EXIT STATUS**

The following exit values are returned:

0  Successful operation.
1  Operation failed.

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWnisu</td>
</tr>
</tbody>
</table>
nischmod(1)

SEE ALSO chmod(1), nis+(1), nischgrp(1), nischown(1), nisdefaults(1), nismatch(1), nis_objects(3NSL), attributes(5)

NOTES NIS+ might not be supported in future releases of the Solaris™ Operating Environment. Tools to aid the migration from NIS+ to LDAP are available in the Solaris 9 operating environment. For more information, visit http://www.sun.com/directory/nisplus/transition.html.
nischown – change the owner of a NIS+ object

nischown [-AfLP] owner name...

nischown changes the owner of the NIS+ objects or entries specified by name to owner. Entries are specified using indexed names (see nismatch(1)). If owner is not a fully qualified NIS+ principal name (see nisaddcred(1M)), the default domain (see nisdefaults(1)) will be appended to it.

The only restriction on changing an object’s owner is that you must have modify permissions for the object. Note: If you are the current owner of an object and you change ownership, you may not be able to regain ownership unless you have modify access to the new object.

The command will fail if the master NIS+ server is not running.

The NIS+ server will check the validity of the name before making the modification.

The following options are supported:

- A	Modify all entries in all tables in the concatenation path that match the search criteria specified in name. It implies the -P option.
- f	Force the operation and fail silently if it does not succeed.
- L	Follow links and change the owner of the linked object or entries rather than the owner of the link itself.
- P	Follow the concatenation path within a named table. This option is only meaningful when either name is an indexed name or the -L option is also specified and the named object is a link pointing to entries.

EXAMPLE 1 Using the nischown Command

The following two examples show how to change the owner of an object to a principal in a different domain, and to change it to a principal in the local domain, respectively.

```
example$ nischown bob.remote.domain. object
example$ nischown skippy object
```

The next example shows how to change the owner of an entry in the passwd table.

```
example$ nischown bob.remote.domain. '[uid=99].passwd.org_dir'
```

This example shows how to change the object or entries pointed to by a link.

```
example$ nischown -L skippy linkname
```

If this variable is set, and the NIS+ name is not fully qualified, each directory specified will be searched until the object is found (see nisdefaults(1)).
nischown(1)

EXIT STATUS
The following exit values are returned:

0 Successful operation.
1 Operation failed.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWnisu</td>
</tr>
</tbody>
</table>

SEE ALSO nis+(1), nischgrp(1), nischmod(1), nischt1(1), nisdefaults(1), nisadcred(1M), nismatch(1), nis_objects(3NSL), attributes(5)

NOTES NIS+ might not be supported in future releases of the Solaris™ Operating Environment. Tools to aid the migration from NIS+ to LDAP are available in the Solaris 9 operating environment. For more information, visit http://www.sun.com/directory/nisplus/transition.html.
**NAME**
nischttl – change the time to live value of a NIS+ object

**SYNOPSIS**
nischttl [-AfLP] time name...

**DESCRIPTION**
nischttl changes the time to live value (ttl) of the NIS+ objects or entries specified by name to time. Entries are specified using indexed names (see nismatch(1)).

The time to live value is used by object caches to expire objects within their cache. When an object is read into the cache, this value is added to the current time in seconds yielding the time when the cached object would expire. The object may be returned from the cache until the current time is earlier than the calculated expiration time. When the expiration time has been reached, the object will be flushed from the cache.

The time to live time may be specified in seconds or in days, hours, minutes, seconds format. The latter format uses a suffix letter of d, h, m, or s to identify the units of time. See the examples below for usage.

The command will fail if the master NIS+ server is not running.

Setting a high ttl value allows objects to stay persistent in caches for a longer period of time and can improve performance. However, when an object changes, in the worst case, the number of seconds in this attribute must pass before that change is visible to all clients. Setting a ttl value of 0 means that the object should not be cached at all.

A high ttl value is a week, a low value is less than a minute. Password entries should have ttl values of about 12 hours (easily allows one password change per day), entries in the RPC table can have ttl values of several weeks (this information is effectively unchanging).

Only directory and group objects are cached in this implementation.

**OPTIONS**
The following options are supported:

- **-A**  Modify all tables in the concatenation path that match the search criterion specified in name. This option implies the -P switch.

- **-f**  Force the operation and fail silently if it does not succeed.

- **-L**  Follow links and change the time to live of the linked object or entries rather than the time to live of the link itself.

- **-P**  Follow the concatenation path within a named table. This option only makes sense when either name is an indexed name or the -L switch is also specified and the named object is a link pointing to entries.

**EXAMPLES**

**EXAMPLE 1** Changing the ttl of an Object

The following example shows how to change the ttl of an object using the seconds format and the days, hours, minutes, seconds format. The ttl of the second object is set to 1 day and 12 hours.
EXAMPLE 1 Changing the ttl of an Object  (Continued)

example% nischttl 184000 object
example% nischttl 1d12h object

EXAMPLE 2 Changing the ttl for a password Entry

This example shows how to change the ttl for a password entry.

example% nischttl 1h30m '[uid=99],passwd.org_dir'

EXAMPLE 3 Changing the ttl of Entries Pointed to by a Link

The next two examples change the ttl of the object or entries pointed to by a link, and the ttl of all entries in the hobbies table.

example% nischttl -L 12h linkname
example% nischttl 3600 '[],hobbies'

ENVIRONMENT VARIABLES

NIS_PATH If this variable is set, and the NIS+ name is not fully qualified, each directory specified will be searched until the object is found. See nisdefaults(1).

EXIT STATUS

The following exit values are returned:

0 Successful operation.
1 Operation failed.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWnisu</td>
</tr>
</tbody>
</table>

SEE ALSO

nis+(1), nischgrp(1), nischmod(1), nischown(1), nisdefaults(1), nismatch(1), nis_objects(3NSL), attributes(5)

NOTES

NIS+ might not be supported in future releases of the Solaris™ Operating Environment. Tools to aid the migration from NIS+ to LDAP are available in the Solaris 9 operating environment. For more information, visit http://www.sun.com/directory/nisplus/transition.html.
nisdefaults(1)

NAME
nisdefaults – display NIS+ default values

SYNOPSIS
nisdefaults [-adghprstv]

DESCRIPTION
The nisdefaults utility prints the default values that are returned by calls to the NIS+ local name functions (see nis_local_names(3NSL)). With no options specified, all defaults will be printed in a verbose format. With options, only that option is displayed in a terse form suitable for shell scripts. See the example below.

OPTIONS
The following options are supported:
- a  Print all defaults in a terse format.
- d  Print the default domain name.
- g  Print the default group name.
- h  Print the default host name.
- p  Print the default principal name.
- r  Print the default access rights with which new objects will be created.
- s  Print the default directory search path.
- t  Print the default time to live value.
- v  Print the defaults in a verbose format. This prepends an identifying string to the output.

EXAMPLES
EXAMPLE 1 Printing NIS+ defaults
The following prints the NIS+ defaults for a root process on machine example in the foo.bar. domain:

eample# nisdefaults
Principal Name : example.foo.bar.
Domain Name : foo.bar.
Host Name : example.foo.bar.
Group Name : .
Access Rights : −−−− −−−− −r−−−−−−
Time to live : 12:00:00
Search Path : foo.bar.

EXAMPLE 2 Setting a variable in the shell script
This example sets a variable in a shell script to the default domain:

DOMAIN='nisdefaults -d'

EXAMPLE 3 Printing the default time to live in verbose format
This example prints out the default time to live in a verbose format:
EXAMPLE 3 Printing the default time to live in verbose format  

```
example% nisdefaults -tv
Time to live : 12:00:00
```

EXAMPLE 4 Printing the time to live in terse format

This example prints out the time to live in the terse format:

```
example% nisdefaults -t
43200
```

Several environment variables affect the defaults associated with a process.

**NIS_DEFAULTS**

This variable contains a defaults string that will override the NIS+ standard defaults. The defaults string is a series of tokens separated by colons. These tokens represent the default values to be used for the generic object properties. All of the legal tokens are described below.

- `ttl=time`
  - This token sets the default time to live for objects that are created. The value `time` is specified in the format as defined by the `nischttl(1)` command. The default value is 12 hours.

- `owner=ownername`
  - This token specifies that the NIS+ principal `ownername` should own created objects. The default for this value is the principal who is executing the command.

- `group=groupname`
  - This token specifies that the group `groupname` should be the group owner for created objects. The default is NULL.

- `access=rights`
  - This token specifies the set of access rights that are to be granted for created objects. The value `rights` is specified in the format as defined by the `nischmod(1)` command. The default value is: `−−−−−−rmcd−−−r−−−−−−−−`.

- **NIS_GROUP**
  - This variable contains the name of the local NIS+ group. If the name is not fully qualified, the default domain will be appended to it.

- **NIS_PATH**
  - This variable overrides the default NIS+ directory search path. It contains an ordered list of directories.
separated by ‘:’ (colon) characters. The ‘$’ (dollar sign) character is treated specially. Directory names that end in ‘$’ have the default domain appended to them, and a ‘$’ by itself is replaced by the list of directories between the default domain and the global root that are at least two levels deep. The default NIS+ directory search path is ‘$’.

Refer to the Name Expansion subsection in nis+(1) for more details.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWnisu</td>
</tr>
</tbody>
</table>

SEE ALSO

nischmod(1), nischttl(1), nis+(1), nis_local_names(3NSL), attributes(5)

NOTES

NIS+ might not be supported in future releases of the Solaris™ Operating Environment. Tools to aid the migration from NIS+ to LDAP are available in the Solaris 9 operating environment. For more information, visit http://www.sun.com/directory/nisplus/transition.html.
niserror(1)

NAME  niserror – display NIS+ error messages

SYNOPSIS  niserror error-num

DESCRIPTION  niserror prints the NIS+ error associated with status value error-num on the standard output. It is used by shell scripts to translate NIS+ error numbers that are returned into text messages.

EXAMPLES  EXAMPLE 1 Using niserror

The following example prints the error associated with the error number 20:

example% niserror 20
Not Found, no such name

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWnisu</td>
</tr>
</tbody>
</table>

SEE ALSO  nis+(1), nis_error(3NSL), attributes(5)

NOTES  NIS+ might not be supported in future releases of the Solaris™ Operating Environment. Tools to aid the migration from NIS+ to LDAP are available in the Solaris 9 operating environment. For more information, visit http://www.sun.com/directory/nisplus/transition.html.
nisgrpadm - NIS+ group administration command

**SYNOPSIS**

```bash
nisgrpadm -a | -r | -t [-s] group principal...
nisgrpadm -d | -l [-M] [-s] group
nisgrpadm -c [-D defaults] [-M] [-s] group
```

**DESCRIPTION**

The `nisgrpadm` utility is used to administer NIS+ groups. This command administers both groups and the groups’ membership lists. `nisgrpadm` can create, destroy, or list NIS+ groups. `nisgrpadm` can be used to administer a group’s membership list. It can add or delete principals to the group, or test principals for membership in the group.

The names of NIS+ groups are syntactically similar to names of NIS+ objects but they occupy a separate namespace. A group named `a.b.c.d.` is represented by a NIS+ group object named `a.groups_dir.b.c.d.``; the functions described here all expect the name of the group, not the name of the corresponding group object.

There are three types of group members:

- An **explicit** member is just a NIS+ principal-name. For example:
  `wickedwitch.west.oz`.

- An **implicit** ("domain") member, written `*.*.west.oz`, means that all principals in the given domain belong to this member. No other forms of wildcarding are allowed; `wickedwitch.*.oz` is invalid, as is `wickedwitch.west.*`. Note that principals in subdomains of the given domain are not included.

- A **recursive** ("group") member, written `@cowards.oz`, refers to another group; all principals that belong to that group are considered to belong here.

Any member may be made **negative** by prefixing it with a minus sign (`-`). A group may thus contain explicit, implicit, recursive, negative explicit, negative implicit, and negative recursive members.

A principal is considered to belong to a group if it belongs to at least one non-negative group member of the group and belongs to no negative group members.

Principal names **must** be fully qualified, whereas groups can be abbreviated on all operations except `create`.

**OPTIONS**

The following options are supported:

- **-a**
  Adds the list of NIS+ principals specified to `group`. The principal name should be fully qualified.

- **-c**
  Creates `group` in the NIS+ namespace. The NIS+ group name should be fully qualified.

- **-d**
  Destroys (removes) `group` from the namespace.

- **-D defaults**
  When creating objects, this option specifies a different set of defaults to be used during this operation. The `defaults` string is a series of tokens separated by colons. These tokens represent the...
default values to be used for the generic object properties. All of
the legal tokens are described below.

`ttl=time` This token sets the default time to
live for objects that are created by
this command. The value `time` is
specified in the format as defined
by the `nischttl(1)` command. The
default value is 12 hours.

`owner=ownername` This token specifies that the NIS+
principal `ownername` should own
the created object. Normally this
value is the same as the principal
who is executing the command.

`group=groupname` This token specifies that the group
`groupname` should be the group
owner for the object that is created.
The default value is `NULL`.

`access=rights` This token specifies the set of access
rights that are to be granted for the
given object. The value `rights` is
specified in the format as defined
by the `nischmod(1)` command. The
default value is 

```
- - - -rmcdcr- - - - - -.
```

`-l` Lists the membership list of the specified `group`. (See `-M` option.)

`-M` Master server only. Sends the lookup to the master server of the
named data. This guarantees that the most up to date information
is seen at the possible expense that the master server may be busy.
Note that the `-M` flag is applicable only with the `-l` flag.

`-r` Removes the list of principals specified from `group`. The principal
name should be fully qualified.

`-s` Work silently. Results are returned using the exit status of the
command. This status can be translated into a text string using the
`niserror(1)` command.

`-t` Displays whether the principals specified are members in `group`.

---

**EXAMPLE 1 Creating a group**

This example shows how to create a group in the `foo.com` domain:

```
exampel% nisgrpadm -c my_buds.foo.com.
```
EXAMPLE 2 How to remove a group
This example shows how to remove the group from the current domain.

```
examp1e% nisgrpadm -d freds_group
```

EXAMPLE 3 Adding to the group
This example shows how one would add two principals, bob and betty, to the group my_buds.foo.com.:

```
```

EXAMPLE 4 How to remove a principal from the group
This example shows how to remove betty from freds_group:

```
examp1e% nisgrpadm -r freds_group betty.foo.com.
```

ENVIRONMENT VARIABLES
NIS_DEFAULTS This variable contains a defaults string that will override the NIS+ standard defaults.
NIS_PATH If this variable is set, and the NIS+ group name is not fully qualified, each directory specified will be searched until the group is found (see nisdefaults(1)).

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWnisu</td>
</tr>
</tbody>
</table>

SEE ALSO nis+(1), nischgrp(1), nischmod(1), nischttl(1), nisdefaults(1), niserror(1), nis_groups(3NSL), attributes(5)

DIAGNOSTICS
NIS_SUCCESS On success, this command returns an exit status of 0.
NIS_PERMISSION When you do not have the needed access right to change the group, the command returns this error.
NIS_NOTFOUND This is returned when the group does not exist.
NIS_TRYAGAIN This error is returned when the server for the group’s domain is currently checkpointing or otherwise in a read-only state. The command should be retried at a later date.
NIS_MODERROR This error is returned when the group was modified by someone else during the execution of the command.
Reissue the command and optionally recheck the group’s membership list.

NOTES
NIS+ might not be supported in future releases of the Solaris™ Operating Environment. Tools to aid the migration from NIS+ to LDAP are available in the Solaris 9 operating environment. For more information, visit http://www.sun.com/directory/nisplus/transition.html.
nisln(1)

NAME
nisln – symbolically link NIS+ objects

SYNOPSIS
nisln [-L] [-D defaults] name linkname

DESCRIPTION
The nisln command links a NIS+ object named name to a NIS+ name linkname. If name is an indexed name (see nismatch(1)), the link points to entries within a NIS+ table. Clients wishing to look up information in the name service can use the FOLLOW_LINKS flag to force the client library to follow links to the name they point to. Further, all of the NIS+ administration commands accept the -L switch indicating they should follow links (see nis_names(3NSL) for a description of the FOLLOW_LINKS flag).

When creating the link, nisln verifies that the linked object exists. Once created, the linked object may be deleted or replaced and the link will not be affected. At that time the link will become invalid and attempts to follow it will return NIS_LINKNAMEERROR to the client. When the path attribute in tables specifies a link rather than another table, the link will be followed if the flag FOLLOW_LINKS was present in the call to nis_list() (see nis_tables(3NSL)) and ignored if the flag is not present. If the flag is present and the link is no longer valid, a warning is sent to the system logger and the link is ignored.

OPTIONS
The following options are supported:

- L
  When present, this option specifies that this command should follow links. If name is itself a link, then this command will follow it to the linked object that it points to. The new link will point to that linked object rather than to name.

- D defaults
  Specify a different set of defaults to be used for the creation of the link object. The defaults string is a series of tokens separated by colons. These tokens represent the default values to be used for the generic object properties. All of the legal tokens are described below.

  ttl=time
  This token sets the default time to live for objects that are created by this command. The value time is specified in the format as defined by the nischttl(1) command. The default is 12 hours.

  owner=ownername
  This token specifies that the NIS+ principal ownername should own the created object. The default for this value is the principal who is executing the command.

  group=groupname
  This token specifies that the group groupname should be the group

User Commands 1017
owner for the object that is created. The default is NULL.

\[ access=rights \]

This token specifies the set of access rights that are to be granted for the given object. The value \textit{rights} is specified in the format as defined by the \texttt{nischmod(1)} command. The default value is \texttt{- - - - rwxdr- - r- - -}.

\section*{Examples}

\subsection*{Example 1 Creating a link}

In this example we create a link in the domain \texttt{foo.com} named \texttt{hosts} that points to the object \texttt{hosts.bar.com}.

\begin{verbatim}
example\% nisln hosts.bar.com. hosts.foo.com.
\end{verbatim}

In this example we make a link \texttt{example.sun.com} that points to an entry in the \texttt{hosts} table in \texttt{eng.sun.com}:

\begin{verbatim}
example\% nisln \( [name=example],hosts.eng.sun.com. \)
\end{verbatim}

\subsection*{Example 2 Making a link that points to an entry in the hosts table}

In this example we make a link \texttt{example.sun.com} that points to an entry in the \texttt{hosts} table in \texttt{eng.sun.com}.

\begin{verbatim}
example\% nisln \( [name=example],hosts.eng.sun.com. \)
\end{verbatim}

\section*{Environment Variables}

\texttt{NIS\_PATH}

If this variable is set, and the NIS+ name is not fully qualified, each directory specified will be searched until the object is found (see \texttt{nisdefaults(1)}).

\section*{Exit Status}

The following exit values are returned:

\begin{verbatim}
0  Successful operation.
1  Operation failed.
\end{verbatim}

\section*{Attributes}

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

\begin{verbatim}
\begin{tabular}{|l|l|}
\hline
ATTRIBUTE TYPE & ATTRIBUTE VALUE \\
\hline
Availability & SUNWnisu \\
\hline
\end{tabular}
\end{verbatim}

\section*{See Also}

nisdefaults(1), nismatch(1), nism(1), nistbladm(1), nis_names(3NSL), nis\_tables(3NSL), attributes(5)

\section*{Notes}

NIS+ might not be supported in future releases of the Solaris\textsuperscript{TM} Operating Environment. Tools to aid the migration from NIS+ to LDAP are available in the Solaris 9 operating environment. For more information, visit http://www.sun.com/directory/nisplus/transition.html.
nisls – list the contents of a NIS+ directory

nisls [-dglLmRM] [name...]

For each name that is a NIS+ directory, nisls lists the contents of the directory. For each name that is a NIS+ object other than a directory, nisls simply echos the name. If no name is specified, the first directory in the search path is listed. See nisdefaults(1).

The following options are supported:

- **-d** Treat NIS+ directories like other NIS+ objects, rather than listing their contents.
- **-g** Display group owner instead of owner when listing in long format.
- **-l** List in long format. This option displays additional information about the objects such as their type, creation time, owner, and access rights. The access rights are listed in the following order in long mode: nobody, owner, group owner, and world.
- **-L** This option specifies that links are to be followed. If name actually points to a link, it is followed to the linked object.
- **-m** Display modification time instead of creation time when listing in long format.
- **-M** Master only. This specifies that information is to be returned from the master server of the named object. This guarantees that the most up to date information is seen at the possible expense that the master server may be busy.
- **-R** List directories recursively. This option will reiterate the list for each subdirectory found in the process of listing each name.

If this variable is set, and the NIS+ name is not fully qualified, each directory specified will be searched until the object is found. See nisdefaults(1).

The following exit values are returned:

0  Successful operation.
1  Operation failed.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWnisu</td>
</tr>
</tbody>
</table>
NIS+ might not be supported in future releases of the Solaris™ Operating Environment. Tools to aid the migration from NIS+ to LDAP are available in the Solaris 9 operating environment. For more information, visit http://www.sun.com/directory/nisplus/transition.html.
### SYNOPSIS

**nismatch** [-AchMoPv] [-s sep] key tablename  
**nismatch** [-AchMoPv] [-s sep] colname = key... tablename  
**nismatch** [-AchMoPv] [-s sep] indexedname  
**nisgrep** [-AchiMov] [-s sep] keypat tablename  
**nisgrep** [-AchiMov] [-s sep] colname = keypat... tablename

### DESCRIPTION

The utilities **nismatch** and **nisgrep** can be used to search NIS+ tables. The command **nisgrep** differs from the **nismatch** command in its ability to accept regular expressions `keypat` for the search criteria rather than simple text matches.

Because **nisgrep** uses a callback function, it is not constrained to searching only those columns that are specifically made searchable at the time of table creation. This makes it more flexible, but slower, than **nismatch**.

In **nismatch**, the server does the searching, whereas in **nisgrep** the server returns all the readable entries and then the client does the pattern-matching.

In both commands, the parameter `tablename` is the NIS+ name of the table to be searched. If only one key or key pattern is specified without the column name, then it is applied searching the first column. Specific named columns can be searched by using the `colname=key` syntax. When multiple columns are searched, only entries that match in all columns are returned. This is the equivalent of a logical join operation.

**nismatch** accepts an additional form of search criteria, `indexedname`, which is a NIS+ indexed name of the form:

```
[colname=value, ...], tablename
```

### OPTIONS

The following options are supported:

- **-A** All data. Return the data within the table and all of the data in tables in the initial table’s concatenation path.
- **-c** Print only a count of the number of entries that matched the search criteria.
- **-h** Display a header line before the matching entries that contains the names of the table’s columns.
- **-i** Ignore upper/lower case distinction during comparisons.
- **-M** Master server only. Send the lookup to the master server of the named data. This guarantees that the most up to date information is seen at the possible expense that the master server may be busy.
- **-o** Display the internal representation of the matching NIS+ object(s).
nismatch(1)

-\*P\*
  Follow concatenation path. Specify that the lookup should follow
  the concatenation path of a table if the initial search is
  unsuccessful.

-\*s sep\*
  This option specifies the character to use to separate the table
  columns. If no character is specified, the default separator for the
  table is used.

-\*v\*
  Verbose. Do not suppress the output of binary data when
  displaying matching entries. Without this option binary data is
  displayed as the string *BINARY*.

EXAMPLE 1 Searching a table for a username

This example searches a table named passwd in the org_dir subdirectory of the
zotz.com domain. It returns the entry that has the username of skippy. In this
example, all the work is done on the server:

example% nismatch name=skippy passwd.org_dir.zotz.com.

EXAMPLE 2 Finding users using specific shells

This example is similar to the one above, except that it uses nisgrep to find all users
in the table named passwd that are using either ksh(1) or csh(1):

example% nisgrep 'shell=[ck]sh' passwd.org_dir.zotz.com.

ENVIRONMENT VARIABLES

NIS_PATH
  If this variable is set, and the NIS+ table name is not
  fully qualified, each directory specified will be searched
  until the table is found (see nisdefaults(1)).

EXIT STATUS

The following exit values are returned:

  0  Successfully matches some entries.
  1  Successfully searches the table and no matches are found.
  2  An error condition occurs. An error message is also printed.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWnisu</td>
</tr>
</tbody>
</table>

SEE ALSO

niscat(1), nisdefaults(1), nisls(1), nistbladm(1), nis_objects(3NSL),
attributes(5)

DIAGNOSTICS

  No memory
  An attempt to allocate some memory for the search failed.
tablename is not a table
   The object with the name tablename was not a table object.

Can’t compile regular expression
   The regular expression in keypat was malformed.

column not found: colname
   The column named colname does not exist in the table named tablename.

NOTES
NIS+ might not be supported in future releases of the Solaris™ Operating Environment. Tools to aid the migration from NIS+ to LDAP are available in the Solaris 9 operating environment. For more information, visit http://www.sun.com/directory/nisplus/transition.html.
nismkdir(1)

NAME nismkdir – create NIS+ directories

SYNOPSIS nismkdir [-D defaults] [-m hostname] [-s hostname] dirnme

DESCRIPTION The nismkdir command creates new NIS+ subdirectories within an existing domain. It can also be used to create replicated directories. Without options, this command will create a subdirectory with the same master and the replicas as its parent directory.

It is advisable to use nisserver(1M) to create an NIS+ domain which consists of the specified directory along with the org_dir and groups_dir subdirectories.

The two primary aspects that are controlled when making a directory are its access rights, and its degree of replication.

A host that serves a NIS+ directory must be a NIS+ client in a directory above the one it is serving. The exceptions to this rule are the root NIS+ servers, which are both clients and servers of the same NIS+ directory.

When the host’s default domain is different from the default domain on the client where the command is executed, the hostname supplied as an argument to the -s or -m options must be fully qualified.

Special per-server and per-directory access restrictions may apply when this command updates the serving lists of the affected NIS+ servers. See nisopaccess(1).

OPTIONS The following options are supported:

- Specify a different set of defaults to be used when creating new directories. The defaults string is a series of tokens separated by colons. These tokens represent the default values to be used for the generic object properties. All of the legal tokens are described below.

  ttl=time This token sets the default time to live for objects that are created by this command. The value time is specified in the format as defined by the nischttl (1) command. The default value is 12h (12 hours).

  owner=ownername This token specifies that the NIS+ principal ownername should own the created object. The default for this value is the principal who is executing the command.

  group=groupname This token specifies that the group groupname should be the group owner for the object that is created. The default value is NULL.
access=rights

This token specifies the set of access rights that are to be granted for the given object. The value rights is specified in the format as defined by the nischmod(1) command. The default value is

- - - -rmcdr- - r- - -.

-m hostname

If the directory named by dirname does not exist, then a new directory that is not replicated is created with host hostname as its master server.

If the directory name by dirname does not exist, then the host named by hostname is made its master server.

-s hostname

Specify that the host hostname will be a replica for an existing directory named dirname.

OPERANDS

The following operand is supported:

dirname The fully qualified NIS+ name of the directory that has to be created.

EXAMPLES

EXAMPLE 1 Using the nismkdir Command

To create a new directory bar under the foo.com. domain that shares the same master and replicas as the foo.com. directory one would use the command:

example% nismkdir bar.foo.com.

To create a new directory bar.foo.com. that is not replicated under the foo.com. domain one would use the command:

example% nismkdir -m myhost.foo.com. bar.foo.com.

To add a replica server of the bar.foo.com. directory, one would use the command:

example% nismkdir -s replica.foo.com. bar.foo.com.

ENVIRONMENT VARIABLES

NIS_DEFAULTS This variable contains a defaults string that will override the NIS+ standard defaults. If the -D switch is used values will then override both the NIS_DEFAULTS variable and the standard defaults.

NIS_PATH If this variable is set, and the NIS+ directory name is not fully qualified, each directory specified will be searched until the directory is found (see nisdefaults(1)).

EXIT STATUS

The following exit values are returned:
nismkdir(1)

0   Successful operation.
1   Operation failed.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWnisu</td>
</tr>
</tbody>
</table>

SEE ALSO nis+(1), nischmod(1), nischt1(1), nisdefaults(1), nisls(1), nisopaccess(1), nisrmdir(1), nisserver(1M), attributes(5)

NOTES NIS+ might not be supported in future releases of the Solaris™ Operating Environment. Tools to aid the migration from NIS+ to LDAP are available in the Solaris 9 operating environment. For more information, visit http://www.sun.com/directory/nisplus/transition.html.
nisopaccess(1)

NAME
nisopaccess – NIS+ operation access control administration command

SYNOPSIS
nisopaccess [-v] directory operation rights
nisopaccess [-v] [-r] directory operation
nisopaccess [-v] [-l] directory [operation]

DESCRIPTION
Most NIS+ operations have implied access control through the permissions on the objects that they manipulate. For example, in order to read an entry in a table, you must have read permission on that entry. However, some NIS+ operations by default perform no access checking at all and are allowed to all:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Example of commands that use the operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIS_CHECKPOINT</td>
<td>nisping -C</td>
</tr>
<tr>
<td>NIS_CPTIME</td>
<td>nisping, rpc.nisd</td>
</tr>
<tr>
<td>NIS_MKDIR</td>
<td>nismkdir</td>
</tr>
<tr>
<td>NIS_PING</td>
<td>nisping, rpc.nisd</td>
</tr>
<tr>
<td>NIS_RMDIR</td>
<td>nisrmkdir</td>
</tr>
<tr>
<td>NIS_SERVSTATE</td>
<td>nisbackup, nisrestore</td>
</tr>
<tr>
<td>NIS_STATUS</td>
<td>nisstat, rpc.nispasswdd</td>
</tr>
</tbody>
</table>

The nisopaccess command can be used to enforce access control on these operations on a per NIS+ directory basis.

The directory argument should be the fully qualified name, including the trailing dot, of the NIS+ directory to which nisopaccess will be applied. As a short-hand, if the directory name does not end in a trailing dot, for example “org_dir”, then the domain name is appended. The domain name is also appended to partial paths such as “org_dir.xyz”.

You can use upper or lower case for the operation argument; however, you cannot mix cases. The “NIS_” prefix may be omitted. For example, NIS_PING can be specified as NIS_PING, nis_ping, PING, or ping.

The rights argument is specified in the format defined by the nischmod(1) command. Since only the read (“r”) rights are used to determine who has the right to perform the operation, the modify and delete rights may be used to control who can change access to the operation.

The access checking performed for each operation is as follows. When an operation requires access be checked on all directories served by its rpc.nisd(1M), access is denied if even one of the directories prohibits the operation.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Check specified directory, or all directories if there is no directory argument (as is the case when NIS_CHECKPOINT is</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIS_CHECKPOINT</td>
<td></td>
</tr>
</tbody>
</table>
NIS_CPTIME
Check specified directory. It returns 0 when access is denied.

NIS_MKDIR
Check parent of specified directory. Returns NIS_PERMISSION when access is denied.

If the parent directory is not available locally, that is, it is not served by this rpc.nisd(1M), NIS_MKDIR access is allowed, though the operation will be executed only if this rpc.nisd is a known replica of the directory.

You should note that the NIS_MKDIR operation does not create a NIS+ directory; it adds a directory to the serving list for this rpc.nisd, if appropriate.

NIS_PING
Check specified directory. No return value.

NIS_RMDIR
Check specified directory. NIS_PERMISSION is returned when access denied.

The NIS_RMDIR operation does not remove a NIS+ directory; it deletes the directory from the serving list for this rpc.nisd, if appropriate.

NIS_SERVSTATE
Check access on all directories served by this rpc.nisd. If access is denied for a tag, "<permission denied>" is returned instead of the tag value.

NIS_STATUS
Same as for NIS_SERVSTATE.

Note that older clients may not supply authentication information for some of the operations listed above. These clients are treated as "nobody" when access checking is performed.

The access control is implemented by creating a NIS+ table called "proto_op_access" in each NIS+ directory to which access control should be applied. The table can be manipulated using normal NIS+ commands. However, nisopaccess is the only supported interface for NIS+ operation access control.

OPTIONS
The following options are supported:

- `l` List the access control for a single operation, or for all operations that have access control enabled.

- `r` Remove access control for a certain operation on the specified directory.

- `v` Verbose mode.
EXAMPLE 1 Enabling Access Control for the NIS_PING Operation

To enable access control for the NIS_PING operation on "org_dir.`domainname`" such that only the owner of the directory can perform a NIS_PING, or change the NIS_PING rights:

```
example% nisopaccess org_dir NIS_PING o=rmcd,g=,w=,n=
```

EXAMPLE 2 Listing the Access to NIS_PING

To list the access to the NIS_PING operation for org_dir:

```
example% nisopaccess -l org_dir NIS_PING
NIS_PING    ----rmcd-------- owner.dom.ain. group.dom.ain.
```

EXAMPLE 3 Removing Access Control for NIS_PING

To remove access control for NIS_PING on org_dir:

```
example% nisopaccess -r org_dir NIS_PING
```

EXIT STATUS

The following exit values are returned:

- 0           Successful operation.
- other       Operation failed. The status is usually the return status from a NIS+ command such as nistbladm.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWnisu</td>
</tr>
</tbody>
</table>

SEE ALSO

nis+(1), nischmod(1), nistbladm(1), rpc.nisd(1M), attributes(5)

NOTES

NIS+ might not be supported in future releases of the Solaris™ Operating Environment. Tools to aid the migration from NIS+ to LDAP are available in the Solaris 9 operating environment. For more information, visit http://www.sun.com/directory/nisplus/transition.html.
The `nispasswd` utility changes a password, gecos (finger) field (-g option), home directory (-h option), or login shell (-s option) associated with the `username` (invoker by default) in the NIS+ passwd table.

Additionally, the command can be used to view or modify aging information associated with the user specified if the invoker has the right NIS+ privileges.

`nispasswd` uses secure RPC to communicate with the NIS+ server, and therefore, never sends unencrypted passwords over the communication medium.

`nispasswd` does not read or modify the local password information stored in the `/etc/passwd` and `/etc/shadow` files.

When used to change a password, `nispasswd` prompts non-privileged users for their old password. It then prompts for the new password twice to forestall typing mistakes. When the old password is entered, `nispasswd` checks to see if it has “aged” sufficiently. If “aging” is insufficient, `nispasswd` terminates; see `getspnam(3C)`.

The old password is used to decrypt the username’s secret key. If the password does not decrypt the secret key, `nispasswd` prompts for the old secure-RPC password. It uses this password to decrypt the secret key. If this fails, it gives the user one more chance. The old password is also used to ensure that the new password differs from the old by at least three characters. Assuming aging is sufficient, a check is made to ensure that the new password meets construction requirements described below. When the new password is entered a second time, the two copies of the new password are compared. If the two copies are not identical, the cycle of prompting for the new password is repeated twice. The new password is used to re-encrypt the user’s secret key. Hence, it also becomes their secure-RPC password. Therefore, the secure-RPC password is no longer a different password from the user’s password.

Passwords must be constructed to meet the following requirements:

- Each password must have at least six characters. Only the first eight characters are significant.
- Each password must contain at least two alphabetic characters and at least one numeric or special character. In this case, “alphabetic” refers to all upper or lower case letters.
- Each password must differ from the user’s login `username` and any reverse or circular shift of that login `username`. For comparison purposes, an upper case letter and its corresponding lowercase letter are equivalent.
New passwords must differ from the old by at least three characters. For comparison purposes, an upper case letter and its corresponding lower case letter are equivalent.

Network administrators, who own the NIS+ password table, may change any password attributes if they establish their credentials (see keylogin(1)) before invoking nispasswd. Hence, nispasswd does not prompt these privileged-users for the old password and they are not forced to comply with password aging and password construction requirements.

Any user may use the -d option to display password attributes for his or her own login name. The format of the display will be:

```
username status mm/dd/yy min max warn
```

or, if password aging information is not present,

```
username status
```

where

- **username** The login ID of the user.
- **status** The password status of **username**: "PS" stands for password exists or locked, "LK" stands for locked, and "NP" stands for no password.
- **mm/dd/yy** The date password was last changed for **username**. (Note that all password aging dates are determined using Greenwich Mean Time (Universal Time) and, therefore, may differ by as much as a day in other time zones.)
- **min** The minimum number of days required between password changes for **username**.
- **max** The maximum number of days the password is valid for **username**.
- **warn** The number of days relative to **max** before the password expires that the **username** will be warned.

The use of nispasswd is strongly discouraged. It is a wrapper around the passwd(1) command.

Using passwd(1) with the -r nisplus option will achieve the same result and will be consistent across all the different name services available. This is the recommended way to change the password in NIS+.

The login program, file access display programs (for example, ls -l), and network programs that require user passwords, for example, rlogin(1), ftp(1), and so on, use the standard getpwnam(3C) and getspnam(3C) interfaces to get password information. These programs will get the NIS+ password information, which is modified by nispasswd, only if the passwd: entry in the /etc/nsswitch.conf file includes nisplus. See nsswitch.conf(4) for more details.
The following options are supported:

- `a` Shows the password attributes for all entries. This will show only the entries in the NIS+ passwd table in the local domain that the invoker is authorized to "read".

- `d [username]` Displays password attributes for the caller or the user specified if the invoker has the right privileges.

- `D domainname` Consults the passwd.org_dir table in domainname. If this option is not specified, the default domainname returned by nis_local_directory() will be used. This domainname is the same as that returned by domainname(1M).

- `f` Forces the user to change password at the next login by expiring the password for username.

- `g` Changes the gecos (finger) information.

- `h` Changes the home directory.

- `l` Locks the password entry for username. Subsequently, login(1) would disallow logins with this NIS+ password entry.

- `n min` Sets minimum field for username. The min field contains the minimum number of days between password changes for username. If min is greater than max, the user may not change the password. Always use this option with the -x option, unless max is set to -1 (aging turned off). In that case, min need not be set.

- `s` Changes the login shell. By default, only the NIS+ administrator can change the login shell. The user will be prompted for the new login shell.

- `w warn` Sets warn field for username. The warn field contains the number of days before the password expires that the user will be warned whenever he or she attempts to login.

- `x max` Sets maximum field for username. The max field contains the number of days that the password is valid for username. The aging for username will be turned off immediately if max is set to -1. If it is set to 0, then the user is forced to change the password at the next login session and aging is turned off.

The following exit values are returned:

0 Success.
1 Permission denied.
2 Invalid combination of options.
3 Unexpected failure. NIS+ passwd table unchanged.
4 NIS+ passwd table missing.
NIS+ is busy. Try again later.
Invalid argument to option.
Aging is disabled.
No memory.
System error.
Account expired.

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWnisu</td>
</tr>
</tbody>
</table>

**SEE ALSO**

keylogin(1), login(1), nis+(1), nistbladm(1), passwd(1), rlogin(1),
domainname(1M), nisserver(1M), getpwnam(3C), getspnam(3C),
nis_local_directory(3NSL), nsswitch.conf(4), passwd(4), shadow(4),
attributes(5)

**NOTES**

NIS+ might not be supported in future releases of the Solaris™ Operating
Environment. Tools to aid the migration from NIS+ to LDAP are available in the
Solaris 9 operating environment. For more information, visit
nisrm(1)

NAME  nisrm – remove NIS+ objects from the namespace

SYNOPSIS  nisrm [-if] name...

DESCRIPTION  The nisrm command removes NIS+ objects named name from the NIS+ namespace. This command will fail if the NIS+ master server is not running.

This command will not remove directories. See nisrmdir(1). Nor will it remove non-empty tables. See nistbladm(1).

OPTIONS  The following options are supported:

  -i     Interactive mode. Like the system rm(1) command the nisrm command will ask for confirmation prior to removing an object. If the name specified by name is a non-fully qualified name this option is forced on. This prevents the removal of unexpected objects.

  -f     Force. The removal is attempted, and if it fails for permission reasons, a nischmod(1) is attempted and the removal retried. If the command fails, it fails silently.

OPERANDS  The following operand is supported:

name     A NIS+ named object.

EXAMPLES  EXAMPLE 1 Using the nisrm Command

Remove the objects foo, bar, and baz from the namespace:

eexample% nisrm foo bar baz

ENVIRONMENT VARIABLES  NIS_PATH     If this variable is set, and the NIS+ name is not fully qualified, each directory specified will be searched until the object is found. See nisdefaults(1).

EXIT STATUS  The following exit values are returned:

  0     Successful operation.

  1     Operation failed.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWnisu</td>
</tr>
</tbody>
</table>

SEE ALSO  nis+(1), nischmod(1), nisdefaults(1), nisrmdir(1), nistbladm(1), rm(1), attributes(5)
NIS+ might not be supported in future releases of the Solaris™ Operating Environment. Tools to aid the migration from NIS+ to LDAP are available in the Solaris 9 operating environment. For more information, visit http://www.sun.com/directory/nisplus/transition.html.
nisrmdir(1)

NAME
nisrmdir – remove NIS+ directories

SYNOPSIS
nisrmdir [-if] [-s hostname] dirnane

DESCRIPTION
nisrmdir deletes existing NIS+ subdirectories. It can remove a directory outright, or
simply remove replicas from serving a directory.

This command modifies the object that describes the directory dirnane, and then
notifies each replica to remove the directory named dirnane. If the notification of any
of the affected replicas fails, the directory object is returned to its original state unless
the -f option is present.

This command will fail if the NIS+ master server is not running.

OPTIONS
The following options are supported:

- i
  Interactive mode. Like the system rm(1) command the nisrmdir command will ask for confirmation prior to removing a directory.
  If the name specified by dirnane is a non-fully qualified name this option is forced on. This prevents the removal of unexpected
directories.

- f
  Force the command to succeed even though it may not be able to
  contact the affected replicas. This option should be used when a
  replica is known to be down and will not be able to respond to the
  removal notification. When the replica is finally rebooted it will
  read the updated directory object, note that it is no longer a replica
  for that directory, and stop responding to lookups on that
directory. Cleanup of the files that held the now removed directory
  can be accomplished manually by removing the appropriate files
  in the /var/nis directory. See nisfiles(4) for more
  information.

- s hostname
  Specify that the host hostname should be removed as a replica for
  the directory named dirnane. If this option is not present all
  replicas and the master server for a directory are removed and the
directory is removed from the namespace.

Special per-server and per-directory access restrictions may apply when this command
updates the serving lists of the affected NIS+ servers. For more information, see
nisopaccess(1).

OPERANDS
The following operand is supported:

dirnane
  An existing NIS+ directory.

EXAMPLES
EXAMPLE 1 Using the nisrmdir Command

To remove a directory bar under the foo.com domain, one would use the
command:

example$ nisrmdir bar.foo.com.
EXAMPLE 1 Using the nisrmdir Command (Continued)

To remove a replica that is serving directory bar.foo.com, one would use the command:

```bash
example% nisrmdir -s replica.foo.com bar.foo.com.
```

To force the removal of directory bar.foo.com from the namespace, one would use the command:

```bash
example% nisrmdir -f bar.foo.com.
```

ENVIRONMENT VARIABLES

NIS_PATH If this variable is set, and the NIS+ directory name is not fully qualified, each directory specified will be searched until the directory is found. See nisdefaults(1).

EXIT STATUS

The following exit values are returned:

- **0**: Successful operation.
- **1**: Operation failed.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWnisu</td>
</tr>
</tbody>
</table>

SEE ALSO

nis+(1), nisdefaults(1), nisopaccess(1), nisrm(1), nisfiles(4), attributes(5)

NOTES

NIS+ might not be supported in future releases of the Solaris™ Operating Environment. Tools to aid the migration from NIS+ to LDAP are available in the Solaris 9 operating environment. For more information, visit http://www.sun.com/directory/nisplus/transition.html.
The nistbladm command is used to administer NIS+ tables. There are five primary operations that it performs: creating and deleting tables, adding entries to, modifying entries within, and removing entries from tables.

Though NIS+ does not place restrictions on the size of tables or entries, the size of data has an impact on the performance and the disk space requirements of the NIS+ server. NIS+ is not designed to store huge pieces of data, such as files; instead, pointers to files should be stored in NIS+.

NIS+ design is optimized to support 10,000 objects with a total size of 10M bytes. If the requirements exceed the above, it is suggested that the domain hierarchy be created, or the data stored in the tables be pointers to the actual data, instead of the data itself.

When creating tables, a table type, type, and a list of column definitions must be provided.

type is a string that is stored in the table and later used by the service to verify that entries being added to it are of the correct type.

Syntax for column definitions is:

colname=[flags][,access]

flags is a combination of:

S Searchable. Specifies that searches can be done on the column’s values (see nismatch(1)).
I Case-insensitive (only makes sense in combination with S). Specifies that searches should ignore case.
C Crypt. Specifies that the column’s values should be encrypted.
nistbladm(1)

B Binary data (does not make sense in combination with S). If not set, the column’s values are expected to be null terminated ASCII strings.

X XDR encoded data (only makes sense in combination with B).

access is specified in the format as defined by the nischmod(1) command.

When manipulating entries, this command takes two forms of entry name. The first uses a series of space separated \texttt{colname=value} pairs that specify column values in the entry. The second is a NIS+ indexed name, \texttt{indexedname}, of the form:

\[ \text{colname=value, . . . }, \text{tablename} \]

\section*{OPTIONS}

The following options are supported:

\begin{itemize}
  \item[-a | A] Adds entries to a NIS+ table. The difference between the lowercase ‘a’ and the uppercase ‘A’ is in the treatment of preexisting entries. The entry’s contents are specified by the \texttt{column=value} pairs on the command line. Values for all columns must be specified when adding entries to a table.

  Normally, NIS+ reports an error if an attempt is made to add an entry to a table that would overwrite an entry that already exists. This prevents multiple parties from adding duplicate entries and having one of them get overwritten. If you wish to force the add, the uppercase ‘A’ specifies that the entry is to be added, even if it already exists. This is analogous to a modify operation on the entry.

  \item[-c] Creates a table named \texttt{tablename} in the namespace. The table that is created must have at least one column and at least one column must be searchable.

  \item[-d \texttt{tablename}] Destroys the table named \texttt{tablename}. The table that is being destroyed must be empty. The table’s contents can be deleted with the \texttt{-R} option below.

  \item[-e | E] Edits the entry in the table that is specified by \texttt{indexedname}. \texttt{indexedname} must uniquely identify a single entry. It is possible to edit the value in a column that would change the indexed name of an entry.

  The change (\texttt{colname=value}) may affect other entries in the table if the change results in an entry whose indexed name is different from \texttt{indexedname} and which matches that of another existing entry. In this case, the \texttt{-e} option will fail and an error will be reported. The \texttt{-E} option will force the replacement of the existing entry by the new entry (effectively removing two old entries and adding a new one).
\end{itemize}
-m A synonym for -E. This option has been superseded by the -E option.

-r | R Removes entries from a table. The xentry is specified by either a series of column=value pairs on the command line, or an indexed name that is specified as entryname. The difference between the interpretation of the lowercase 'r' versus the uppercase 'R' is in the treatment of non-unique entry specifications. Normally the NIS+ server will disallow an attempt to remove an entry when the search criterion specified for that entry resolves to more than one entry in the table. However, it is sometimes desirable to remove more than one entry, as when you are attempting to remove all of the entries from a table. In this case, using the uppercase 'R' will force the NIS+ server to remove all entries matching the passed search criterion. If that criterion is null and no column values specified, then all entries in the table will be removed.

-u Updates attributes of a table. This allows the concatenation path (-p), separation character (specified with the (-s)), column access rights, and table type string (-t) of a table to be changed. Neither the number of columns, nor the columns that are searchable may be changed.

-D defaults When creating objects, this option specifies a different set of defaults to be used during this operation. The defaults string is a series of tokens separated by colons. These tokens represent the default values to be used for the generic object properties. All of the legal tokens are described below.

ttl=time This token sets the default time to live for objects that are created by this command. The value time is specified in the format as defined by the nischttl(1) command. The default value is 12 hours.

owner=ownername This token specifies that the NIS+ principal ownername should own the created object. Normally this value is the same as the principal who is executing the command.

group=groupname This token specifies that the group groupname should be the group owner for the object that is created. The default value is NULL.

access=rights This token specifies the set of access rights that are to be granted for the given object. The value rights is
The column name has read access for all (that is, owner, group, and world) and modify access for only the owner. The column hobby is readable by all, but not modifiable by anyone.

In this example, if the access rights had not been specified, the table's access rights would have come from either the standard defaults or the NIS_DEFAULTS variable (see below).

**EXAMPLE 2 Adding Entries to the Table**

To add entries to this table:

```
example$ nistbladm -a name=bob hobby=skiing hobbies.foo.com.
example$ nistbladm -a name=sue hobby=skiing hobbies.foo.com.
example$ nistbladm -a name=ted hobby=swimming hobbies.foo.com.
```

**EXAMPLE 3 Adding the Concatenation Path**

In the following example, the common root domain is foo.com (NIS+ requires at least two components to define the root domain) and the concatenation path for the subdomains bar and baz are added:
EXAMPLE 3 Adding the Concatenation Path

Example:
```
nistbladm -u -p hobbies.bar.foo.com:hobbies.baz.foo.com. hobbies
```

EXAMPLE 4 Deleting Skiers from the List

To delete the skiers from our list:
```
nistbladm -R hobby=skiing hobbies.foo.com.
```

Note: The use of the `-r` option would fail because there are two entries with the value of `skiing`.

EXAMPLE 5 Naming a Column with no Flags Set

To create a table with a column that is named with no flags set, you supply only the name and the equals (=) sign as follows:
```
nistbladm -c notes_tbl name=S,a+r,o+m note= notes.foo.com.
```

This example created a table, named `notes.foo.com`, of type `notes_tbl` with two columns `name` and `note`. The `note` column is not searchable.

EXAMPLE 6 Protecting Terminal Characters

When entering data for columns in the form of a `value string`, it is essential that terminal characters be protected by single or double quotes. These are the characters equals (=), comma (,), left bracket ([), right bracket (]), and space ( ). These characters are parsed by NIS+ within an indexed name. These characters are protected by enclosing the entire value in double quote (" ) characters as follows:
```
nistbladm -a fullname="Joe User" nickname=Joe nicknames
```

If there is any doubt about how the string will be parsed, it is better to enclose it in quotes.

ENVIRONMENT VARIABLES

- **NIS_DEFAULTS**: This variable contains a defaults string that will override the NIS+ standard defaults. If the `-D` switch is used those values will then override both the `NIS_DEFAULTS` variable and the standard defaults.

- **NIS_PATH**: If this variable is set, and the NIS+ table name is not fully qualified, each directory specified will be searched until the table is found. See `nisdefaults(1)`.

EXIT STATUS

The following exit values are returned:
Successful operation.

Operation failed.

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWnisu</td>
</tr>
</tbody>
</table>

**SEE ALSO**

nis+(1), niscat(1), nischmod(1), nischown(1), nischt1(1), nisdefaults(1), nismatch(1), nissetup(1M), attributes(5)

**NOTES**

NIS+ might not be supported in future releases of the Solaris™ Operating Environment. Tools to aid the migration from NIS+ to LDAP are available in the Solaris 9 operating environment. For more information, visit http://www.sun.com/directory/nisplus/transition.html.

**WARNINGS**

To modify one of the entries, say, for example, from “bob” to “robert”:

```bash
example% nistbladm -m name=robert [name=bob],hobbies
```

Notice that “[name=bob],hobbies” is an indexed name, and that the characters ‘[’ (open bracket) and ‘]’ (close bracket) are interpreted by the shell. When typing entry names in the form of NIS+ indexed names, the name must be protected by using single quotes.

It is possible to specify a set of defaults such that you cannot read or modify the table object later.
nigest(1)

NAME  nistest – return the state of the NIS+ namespace using a conditional expression

SYNOPSIS  nistest [-ALMP] [-a rights | -t type] object
           nistest [-ALMP] [-a rights] indexedname
           nistest -c dir1 op dir2

DESCRIPTION  nistest provides a way for shell scripts and other programs to test for the existence, type, and access rights of objects and entries. Entries are named using indexed names. See nismatch(1). With the -c option, directory names can be compared to test where they lie in relation to each other in the namespace.

OPTIONS  The following options are supported:

- a rights  This option is used to verify that the current process has the desired or required access rights on the named object or entries. The access rights are specified in the same way as the nischmod(1) command.

-A  All data. This option specifies that the data within the table and all of the data in tables in the initial table’s concatenation path be returned. This option is only valid when using indexed names or following links.

-L  Follow links. If the object named by object or the tablename component of indexedname names a LINK type object, the link is followed when this switch is present.

-M  Master server only. This option specifies that the lookup should be sent to the master server of the named data. This guarantees that the most up to date information is seen at the possible expense that the master server may be busy.

-P  Follow concatenation path. This option specifies that the lookup should follow the concatenation path of a table if the initial search is unsuccessful. This option is only valid when using indexed names or following links.

-t type  This option tests the type of object. The value of type can be one of the following:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>Return true if the object is a directory object.</td>
</tr>
<tr>
<td>G</td>
<td>Return true if the object is a group object.</td>
</tr>
<tr>
<td>L</td>
<td>Return true if the object is a link object.</td>
</tr>
<tr>
<td>P</td>
<td>Return true if the object is a private object.</td>
</tr>
<tr>
<td>T</td>
<td>Return true if the object is a table object.</td>
</tr>
</tbody>
</table>

-c  Test whether or not two directory names have a certain relationship to each other, for example, higher than (ht) or lower than (lt). The complete list of values for op can be displayed by
using the -c option with no arguments.

EXAMPLE 1 Using the nistest Command

When testing for access rights, nistest returns success (0) if the specified rights are granted to the current user. Thus, testing for access rights:

```
example$ nistest -a w=mr skippy.domain
```

Tests that all authenticated NIS+ clients have read and modify access to the object named `skippy.domain`.

Testing for access on a particular entry in a table can be accomplished using the indexed name syntax. The following example tests to see if an entry in the password table can be modified:

```
example$ nistest -a o=m '[uid=99],passwd.org_dir'
```

To test if a directory lies higher in the namespace than another directory, use the -c option with an op of ht (higher than) as in the following example (which would return true):

```
```

NIS_PATH
If this variable is set, and the NIS+ name is not fully qualified, each directory specified will be searched until the object is found. See nisdefaults(1).

EXIT STATUS
The following exit values are returned:

- **0**: Successful operation.
- **1**: Failure due to object not present, not of specified type, and/or no such access.
- **2**: Failure due to illegal usage.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWnisu</td>
</tr>
</tbody>
</table>

SEE ALSO
nis+(1), nischmod(1), nisdefaults(1), nismatch(1), attributes(5)

NOTES
NIS+ might not be supported in future releases of the Solaris™ Operating Environment. Tools to aid the migration from NIS+ to LDAP are available in the Solaris 9 operating environment. For more information, visit http://www.sun.com/directory/nisplus/transition.html.
The `nl` command reads lines from the named file, or the standard input if no file is named, and reproduces the lines on the standard output. Lines are numbered on the left in accordance with the command options in effect.

`nl` views the text it reads in terms of logical pages. Line numbering is reset at the start of each logical page. A logical page consists of a header, a body, and a footer section. Empty sections are valid. Different line numbering options are independently available for header, body, and footer. For example, `-bt` (the default) numbers non-blank lines in the body section and does not number any lines in the header and footer sections.

The start of logical page sections are signaled by input lines containing nothing but the following delimiter character(s):

```
\:\:\:  header
\:\:\  body
\:\  footer
```

Unless optioned otherwise, `nl` assumes the text being read is in a single logical page body.

**OPTIONS**

Command options may appear in any order and may be intermingled with an optional file name. Only one file may be named. The specified default is used when the option is not entered on the command line. `/usr/xpg4/bin/nl` options require option arguments. A SPACE character _may_ separate options from option arguments. `/usr/bin/nl` options _may_ have option arguments. If option-arguments of `/usr/bin/nl` options are not specified, these options result in the default. The supported options are:

- `-btype`
  
  Specifies which logical page body lines are to be numbered.

  Recognized _types_ and their meanings are:

  - `a` number all lines
  - `t` number all non-empty lines.
  - `n` no line numbering
The following operand is supported:

file  A path name of a text file to be line-numbered.

OPERANDS

The following operand is supported:

file  A path name of a text file to be line-numbered.
EXAMPLE 1 An example of the nl command.

The command:

```
example% nl -v10 -i10 -d!+ filename1
```

will cause the first line of the page body to be numbered 10, the second line of the page body to be numbered 20, the third 30, and so forth. The logical page delimiters are !+.

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of nl: LC_COLLATE, LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS

The following exit values are returned:

<table>
<thead>
<tr>
<th>Exit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>&gt;0</td>
<td>An error occurred.</td>
</tr>
</tbody>
</table>

FILES

```
/usr/lib/locale/locale/LC_COLLATE/CollTable
```

collision table generated by localedef

```
/usr/lib/locale/locale/LC_COLLATE/coll.so
```

shared object containing string transformation library routines

ATTRIBUTES

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

```
ATTRIBUTES FOR/usr/bin/nl

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
</tbody>
</table>
```

```
ATTRIBUTES FOR/usr/xpg4/bin/nl

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWxcu4</td>
</tr>
</tbody>
</table>
```

SEE ALSO

pr(1), attributes(5), environ(5), regex(5), regexp(5)

NOTES

Internationalized Regular Expressions are used in the POSIX and "C" locales. In other locales, Internationalized Regular Expressions are used if the following two conditions are met:

- /usr/lib/locale/locale/LC_COLLATE/CollTable is present
- /usr/lib/locale/locale/LC_COLLATE/coll.so is not present;

otherwise, Simple Regular Expressions are used.

Internationalized Regular Expressions are explained on regex(5). Simple Regular Expressions are explained on regexp(5).
NAME

nm - print name list of an object file

SYNOPSIS

/usr/ccs/bin/nm [-ACdhlnPprRsTuVv] [-efo] [-g | -u] [-t format] file...

/usr/xpg4/bin/nm [-ACdhlnPprRsTuVv] [-efo] [-g | -u] [-t format] file...

DESCRIPTION

The nm utility displays the symbol table of each ELF object file that is specified by file.

If no symbolic information is available for a valid input file, the nm utility will report that fact, but not consider it an error condition.

OPTIONS

The output of nm may be controlled using the following options:

- A
  Writes the full path name or library name of an object on each line.

- C
  Demangles C++ symbol names before printing them out.

- D
  Displays the SHT_DYNSYM symbol information. This is the symbol table used by ld.so.1 and is present even in stripped dynamic executables. By default, the SHT_SYMTAB symbol table is displayed.

- e
  See NOTES below.

- f
  See NOTES below.

- g
  Writes only external (global) symbol information.

- h
  Does not display the output heading data.

- l
  Distinguishes between WEAK and GLOBAL symbols by appending a * to the key letter for WEAK symbols.

- n
  Sorts external symbols by name before they are printed.

- o
  Prints the value and size of a symbol in octal instead of decimal (equivalent to -t o).

- p
  Produces easy to parse, terse output. Each symbol name is preceded by its value (blanks if undefined) and one of the letters:

A Absolute symbol.
B bss (uninitialized data space) symbol.
C COMMON symbol.
D Data object symbol.
F File symbol.
N Symbol has no type.
L Thread-Local storage symbol.
S Section symbol.
T   Text symbol.

U   Undefined.

If the symbol’s binding attribute is:

LOCAL   The key letter is lower case.

WEAK    The key letter is upper case. If the -1 modifier
         is specified, the upper case key letter is
         followed by a *

GLOBAL   The key letter is upper case.

-P   Writes information in a portable output format, as specified in
      Standard Output.

-r   Prepends the name of the object file or archive to each output line.

-R   Prints the archive name (if present), followed by the object file and
      symbol name. If the -r option is also specified, this option is
      ignored.

-s   Prints section name instead of section index.

-t format   Writes each numeric value in the specified format. The format is
             dependent on the single character used as the format
             option-argument:
             d   The offset is written in decimal (default).
             o   The offset is written in octal.
             x   The offset is written in hexadecimal.

-T   See NOTES below.

/usr/ccs/bin/nm   -u   Prints undefined symbols only.

/usr/xpg4/bin/nm  -u   Prints long listing for each undefined symbol. See OUTPUT below.

-v   Sorts external symbols by value before they are printed.

-V   Prints the version of the nm command executing on the standard
      error output.

-x   Prints the value and size of a symbol in hexadecimal instead of
      decimal (equivalent to -t x).

Options may be used in any order, either singly or in combination, and may appear
anywhere in the command line. When conflicting options are specified (such as -v
and -n, or -o and -x) the first is taken and the second ignored with a warning
message to the user. (See -R for exception.)

OPERANDS

The following operand is supported:

file   A path name of an object file, executable file or object-file library.
**Standard Output**

For each symbol, the following information will be printed:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Index</strong></td>
<td>The index of the symbol. (The index appears in brackets.)</td>
</tr>
<tr>
<td><strong>Value</strong></td>
<td>The value of the symbol is one of the following:</td>
</tr>
<tr>
<td></td>
<td>- A section offset for defined symbols in a relocatable file.</td>
</tr>
<tr>
<td></td>
<td>- Alignment constraints for symbols whose section index is <code>SHN_CLASS</code>.</td>
</tr>
<tr>
<td></td>
<td>- A virtual address in executable and dynamic library files.</td>
</tr>
<tr>
<td><strong>Size</strong></td>
<td>The size in bytes of the associated object.</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>A symbol is of one of the following types:</td>
</tr>
<tr>
<td></td>
<td>- <strong>NOTYPE</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>OBJECT</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>FUNC</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>REGI</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>SECTION</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>FILE</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>COMMON</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>TLS</strong></td>
</tr>
<tr>
<td><strong>Bind</strong></td>
<td>The symbol’s binding attributes.</td>
</tr>
<tr>
<td></td>
<td>- <strong>LOCAL symbols</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>GLOBAL symbols</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>WEAK symbols</strong></td>
</tr>
<tr>
<td><strong>Other</strong></td>
<td>A field reserved for future use, currently containing 0.</td>
</tr>
<tr>
<td><strong>Shndx</strong></td>
<td>Except for three special values, this is the section header table index in relation to which the symbol is defined. The following special values exist:</td>
</tr>
<tr>
<td></td>
<td>- <strong>ABS</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>COMMON</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>UNDEF</strong></td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>The name of the symbol.</td>
</tr>
</tbody>
</table>
Object Name  The name of the object or library if -A is specified.

If the -P option is specified, the previous information is displayed using the following portable format. The three versions differ depending on whether -t d, -t o, or -t x was specified, respectively:

"%s%s %d %d\n", <library/object name>, name, type, value, size 
"%s%s %o %o\n", <library/object name>, name, type, value, size 
"%s%s %x %x\n", <library/object name>, name, type, value, size

where <library/object name> is formatted as follows:

- If -A is not specified, <library/object name> is an empty string.
- If -A is specified and the corresponding file operand does not name a library:
  "%s: ", file
- If -A is specified and the corresponding file operand names a library. In this case, <object file> names the object file in the library containing the symbol being described:
  "%s[%s]: ", file, <object file>

If -A is not specified, then if more than one file operand is specified or if only one file operand is specified and it names a library, nm will write a line identifying the object containing the following symbols before the lines containing those symbols, in the form:

- If the corresponding file operand does not name a library:
  "%s:\n", file
- If the corresponding file operand names a library; in this case, <object file> is the name of the file in the library containing the following symbols:
  "%s[%s]:\n", file, <object file>

If -P is specified, but -t is not, the format is as if -t x had been specified.

ENVIRONMENT VARIABLES
See environ(5) for descriptions of the following environment variables that affect the execution of nm: LC_COLLATE, LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS
The following exit values are returned:
0       Successful completion.
>0      An error occurred.

ATTRIBUTES
See attributes(5) for descriptions of the following attributes:
The following options are obsolete because of changes to the object file format and will be deleted in a future release.

- `e` Prints only external and static symbols. The symbol table now contains only static and external symbols. Automatic symbols no longer appear in the symbol table. They do appear in the debugging information produced by `cc -g`, which may be examined using `dump(1)`.

- `f` Produces full output. Redundant symbols (such as `.text`, `.data`, and so forth), which existed previously, do not exist and producing full output will be identical to the default output.

- `T` By default, `nm` prints the entire name of the symbols listed. Since symbol names have been moved to the last column, the problem of overflow is removed and it is no longer necessary to truncate the symbol name.

SEE ALSO

`ar(1), as(1), dump(1), ld(1), ld.so.1(1), ar(3HEAD), a.out(4), attributes(5), environ(5), XPG4(5)`
The `nohup` utility invokes the named `command` with the arguments supplied. When the `command` is invoked, `nohup` arranges for the `SIGHUP` signal to be ignored by the process.

When invoked with the `-p` or `-g` flags, `nohup` arranges for processes already running as identified by a list of process IDs or a list of process group IDs to become immune to hangups.

The `nohup` utility can be used when it is known that `command` will take a long time to run and the user wants to log out of the terminal. When a shell exits, the system sends its children `SIGHUP` signals, which by default cause them to be killed. All stopped, running, and background jobs will ignore `SIGHUP` and continue running, if their invocation is preceded by the `nohup` command or if the process programmatically has chosen to ignore `SIGHUP`.

Processes run by `/usr/bin/nohup` are immune to `SIGHUP` (hangup) and `SIGQUIT` (quit) signals.

Processes specified by ID are made immune to `SIGHUP` and `SIGQUIT`, and all output to the controlling terminal is redirected to `nohup.out`. If `-F` is specified, `nohup` will force control of each process. If `-a` is specified, `nohup` will change the signal disposition of `SIGHUP` and `SIGQUIT` even if the process has installed a handler for either signal.

Every process in the same process group as the processes specified by ID are made immune to `SIGHUP` and `SIGQUIT`, and all output to the controlling terminal is redirected to `nohup.out`. If `-F` is specified, `nohup` will force control of each process. If `-a` is specified, `nohup` will change the signal disposition of `SIGHUP` and `SIGQUIT` even if the process has installed a handler for either signal.

Processes run by `/usr/xpg4/bin/nohup` are immune to `SIGHUP`.

The `nohup` utility does not arrange to make processes immune to a `SIGTERM` (terminate) signal, so unless they arrange to be immune to `SIGTERM` or the shell makes them immune to `SIGTERM`, they will receive it.
If `nohup.out` is not writable in the current directory, output is redirected to
`$HOME/nohup.out`. If a file is created, the file will have read and write permission
(600, see `chmod(1)`). If the standard error is a terminal, it is redirected to the
standard output, otherwise it is not redirected. The priority of the process run by
`nohup` is not altered.

**OPTIONS**
The following options are supported:

- `-a` Always changes the signal disposition of target processes. This option is
valid only when specified with `-p` or `-g`.
- `-F` Force. Grabs the target processes even if another process has control. This
option is valid only when specified with `-p` or `-g`.
- `-g` Operates on a list of process groups. This option is not valid with `-p`.
- `-p` Operates on a list of processes. This option is not valid with `-g`.

**OPERANDS**
The following operands are supported:

- `pid` A decimal process ID to be manipulated by `nohup -p`.
- `pgid` A decimal process group ID to be manipulated by `nohup -g`.
- `command` The name of a command that is to be invoked. If the `command`
operand names any of the special `shell_builtins(1)` utilities, the results are undefined.
- `argument` Any string to be supplied as an argument when invoking the
`command` operand.

**EXAMPLES**
**EXAMPLE 1** Applying `nohup` to pipelines or command lists

It is frequently desirable to apply `nohup` to pipelines or lists of commands. This can be
done only by placing pipelines and command lists in a single file, called a shell script.
One can then issue:

```
example$ nohup sh file
```

and the `nohup` applies to everything in `file`. If the shell script `file` is to be executed
often, then the need to type `sh` can be eliminated by giving `file` execute permission.

Add an ampersand and the contents of `file` are run in the background with interrupts
also ignored (see `sh(1)`):

```
example$ nohup file &
```

**EXAMPLE 2** Applying `nohup -p` to a process

```
exanple$ long_running_command &
exanple$ nohup -p 'pgrep long_running_command'
```
EXAMPLE 3 Applying nohup -g to a process group

```bash
example$ make &
example$ ps -o sid -p $$
SID 81079
example$ nohup -g 'pgrep -s 81079 make'
```

**ENVIRONMENT VARIABLES**

See environ(5) for descriptions of the following environment variables that affect the execution of nohup: LC_CTYPE, LC_MESSAGES, PATH, and NLSPATH.

- **HOME**
  - Determine the path name of the user's home directory: if the output file nohup.out cannot be created in the current directory, the nohup command will use the directory named by HOME to create the file.

**EXIT STATUS**

The following exit values are returned:

- **126** command was found but could not be invoked.
- **127** An error occurred in nohup, or command could not be found

Otherwise, the exit values of nohup will be those of the command operand.

**FILES**

- **nohup.out**
  - The output file of the nohup execution if standard output is a terminal and if the current directory is writable.

- **$HOME/nohup.out**
  - The output file of the nohup execution if standard output is a terminal and if the current directory is not writable.

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>/usr/bin/nohup</th>
<th>/usr/xpg4/bin/nohup</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ATTRIBUTE TYPE</strong></td>
<td><strong>ATTRIBUTE VALUE</strong></td>
</tr>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ATTRIBUTE TYPE</strong></td>
<td><strong>ATTRIBUTE VALUE</strong></td>
</tr>
<tr>
<td>Availability</td>
<td>SUNWxcu4</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

**SEE ALSO**

batch(1), chmod(1), csh(1), ksh(1), nice(1), pgrep(1), proc(1), ps(1), sh(1),
shell_builtins(1), signal(3C), proc(4), attributes(5), environ(5), XPG4(5)

**WARNINGS**

If you are running the Korn shell (ksh(1)) as your login shell, and have nohup'ed jobs running when you attempt to log out, you will be warned with the message
You have jobs running.

You will then need to log out a second time to actually log out. However, your background jobs will continue to run.

The C-shell (csh(1)) has a built-in command nohup that provides immunity from SIGHUP, but does not redirect output to nohup.out. Commands executed with '&\n' are automatically immune to HUP signals while in the background.

nohup does not recognize command sequences. In the case of the following command,

```
example$ nohup command1; command2
```

the nohup utility applies only to command1. The command,

```
example$ nohup (command1; command2)
```

is syntactically incorrect.
nroff(1)

NAME
nroff – format documents for display or line-printer

SYNOPSIS
nroff [-ehiq] [-m name] [-nN] [-opagelist] [-raN] [-sN] [-Tname]
[-uN]

DESCRIPTION
The nroff utility formats text in the named files for typewriter-like devices. See also troff(1).

If no file argument is present, nroff reads the standard input. An argument consisting of a ‘−’ is taken to be a file name corresponding to the standard input.

OPTIONS
The following options are supported. Options may appear in any order so long as they appear before the files.

-e Produces equally-spaced words in adjusted lines, using full terminal resolution.

-h Uses output TAB characters during horizontal spacing to speed output and reduces output character count. TAB settings are assumed to be every 8 nominal character widths.

-i Reads the standard input after the input files are exhausted.

-q Does not print output that was read from an .rd request.

-m name Prepends the macro file /usr/share/lib/tmac/tmac.name to the input files.

-nN Numbers first generated page N.

-opagelist Prints only pages whose page numbers appear in the comma-separated list of numbers and ranges. A range N–M means pages N through M; an initial −N means from the beginning to page N; and a final N− means from N to the end.

-raN Sets register a (one-character) to N.

-sN Stops every N pages. nroff will halt prior to every N pages (default N=1) to allow paper loading or changing, and will resume upon receipt of a NEWLINE.

-Tname Prepares output for a device of the specified name. Known names are:

37 Teletype Corporation Model 37 terminal — this is the default.

1p l tn300 GE — any line printer or terminal without half-line capability.

300 DASI-300.

300-12 DASI-300 — 12-pitch.

300S DASI-300S.
300S-12  DASI-300S.
382  DASI-382 (fancy DTC 382).
450  DASI-450 (Diablo Hyterm).
450-12  DASI-450 (Diablo Hyterm) — 12-pitch.
832  AJ 832.

-uN  Set the emboldening factor for the font mounted in position 3 to N.
     If N is missing, then set the emboldening factor to 0.

EXAMPLE 1  Formatting with a macro package

The following command formats users.guide using the -me macro package, and
stopping every 4 pages:

    example% nroff -s4 -me users.guide

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the
execution of nroff: LC_CTYPE, LC_MESSAGES, and NLSPATH.

FILES

    /usr/tmp/trtmp*  temporary file (see NOTES)
    /usr/share/lib/tmac/tmac.*  standard macro files
    /usr/share/lib/nterm/*  terminal driving tables for nroff
    /usr/share/lib/nterm/README  index to terminal description files

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWdoc</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

SEE ALSO

checknr(1), col(1), eqn(1), man(1), tbl(1), troff(1), attributes(5), environ(5),
me(5), ms(5), term(5)

NOTES

/usr/tmp is currently a symbolic link to /var/tmp.

Previous documentation incorrectly described the numeric register yr as being the
"Last two digits of current year". yr is in actuality the number of years since 1900. To
correctly obtain the last two digits of the current year through the year 2099, the
definition given below of string register yy may be included in a document and
subsequently used to display a two-digit year. Notice that any other available one- or
two-character register name may be substituted for yy.

."  definition of new string register yy--last two digits of year
."  yr (# of years since 1900) if it is < 100
nroff(1)

.ie \n{yr<100 .ds yy \n
.el \} ." else, subtract 100 from yr, store in ny
.nr ny \n\{yr-100
.ie \n\{ny>9 \{ ." use ny if it is two digits
.ds yy \n\{ny
.\" remove temporary number register ny
.rr ny \}
.el \}{.ds yy 0
.\" if ny is one digit, append it to 0
.as yy \n\{ny
.rr ny \} \} \}
NAME
od – octal dump

SYNOPSIS
/usr/bin/od [-bcCDdFfOoSsvXx] [-] [file] [offset_string]
/usr/bin/od [-bcCDdFfOoSsvXx] [-A address_base] [-j skip] [-N count]
[-t type_string...] [-] [file...]
/usr/xpg4/bin/od [-bcCDdFfOoSsvXx] [-] [file] [offset_string]
/usr/xpg4/bin/od [-bcCDdFfOoSsvXx] [-A address_base] [-j skip]
[-N count] [-t type_string...] [-] [file...]

DESCRIPTION
The od command copies sequentially each input file to standard output and transforms the input data according to the output types specified by the -t or -bcCdFfOoSsvXx options. If no output type is specified, the default output is as if -t oo2 had been specified. Multiple types can be specified by using multiple -bcCdFfOoSsvXx options. Output lines are written for each type specified in the order in which the types are specified. If no file is specified, the standard input is used.

The [offset_string] operand is mutually exclusive from the -A, -j, -N, and -t options.

For the purposes of this description, the following terms are used:

word Refers to a 16-bit unit, independent of the word size of the machine.
long word Refers to a 32-bit unit.
double long word Refers to a 64-bit unit.

OPTIONS
The following options are supported:

-A address_base Specify the input offset base. The address_base option-argument must be a character. The characters d, o and x specify that the offset base will be written in decimal, octal or hexadecimal, respectively. The character n specifies that the offset will not be written. Unless -An is specified, the output line will be preceded by the input offset, cumulative across input files, of the next byte to be written. In addition, the offset of the byte following the last byte written will be displayed after all the input data has been processed. Without the -A address_base option and the [offset_string] operand, the input offset base is displayed in octal.

-b Interpret bytes in octal. This is equivalent to -t oo1.

-c Display single-byte characters. Certain non-graphic characters appear as C-language escapes:

null \0
backspace \b
form-feed \f
new-line \n
return \r
tab \t

Others appear as 3-digit octal numbers. For example:
-c  Interpret bytes as single-byte or multibyte characters according to
    the current setting of the LC_CTYPE locale category.  Printable
    multibyte characters are written in the area corresponding to the
    first byte of the character; the two character sequence ** is written
    in the area corresponding to each remaining byte in the character,
    as an indication that the character is continued.  Non-graphic
    characters appear the same as they would using the -C option.

-C  Interpret bytes as single-byte or multibyte characters according to
    the current setting of the LC_CTYPE locale category.  Printable
    multibyte characters are written in the area corresponding to the
    first byte of the character; two character sequence ** are written in
    the area corresponding to each remaining byte in the character, as
    an indication that the character is continued.  Certain non-graphic
    characters appear as C escapes:

    null      \0
    backspace \b
    form-feed \f
    new-line  \n
    return    \r
    tab       \t

    Other non-printable characters appear as one three-digit octal
    number for each byte in the character.

-d  Interpret words in unsigned decimal.  This is equivalent to -t u2.

-D  Interpret long words in unsigned decimal.  This is equivalent to -t u4.

-f  Interpret long words in floating point.  This is equivalent to -t f4.

-F  Interpret double long words in extended precision.  This is
    equivalent to -t f8.

- j skip  Jump over skip bytes from the beginning of the input.  The od
    command will read or seek past the first skip bytes in the
    concatenated input files.  If the combined input is not at least skip
    bytes long, the od command will write a diagnostic message to
    standard error and exit with a non-zero exit status.

    By default, the skip option-argument is interpreted as a decimal
    number.  With a leading 0x or 0X, the offset is interpreted as a
    hexadecimal number; otherwise, with a leading 0, the offset will be
    interpreted as an octal number.  Appending the character b, k, or m
    to offset will cause it to be interpreted as a multiple of 512, 1024
    or 1 048 576 bytes, respectively.  If the skip number is
hexadecimal, any appended \( b \) is considered to be the final hexadecimal digit. The address is displayed starting at \( 0000000 \), and its base is not implied by the base of the \( \text{skip} \) option-argument.

\(-N \text{ count} \)

Format no more than \textit{count} bytes of input. By default, \textit{count} is interpreted as a decimal number. With a leading \( 0x \) or \( OX \), \textit{count} is interpreted as a hexadecimal number; otherwise, with a leading \( 0 \), it is interpreted as an octal number. If \textit{count} bytes of input (after successfully skipping, if \(-j \text{skip} \) is specified) are not available, it will not be considered an error; the \texttt{od} command will format the input that is available. The base of the address displayed is not implied by the base of the \textit{count} option-argument.

\(-o\)

Interpret words in octal. This is equivalent to \(-t \text o2\).

\(-O\)

Interpret long words in unsigned octal. This is equivalent to \(-t \text o4\).

\(-s\)

Interpret words in signed decimal. This is equivalent to \(-t \text d2\).

\(-S\)

Interpret long words in signed decimal. This is equivalent to \(-t \text d4\).

\(-t \text{ type_string} \)

Specify one or more output types. The \textit{type_string} option-argument must be a string specifying the types to be used when writing the input data. The string must consist of the type specification characters:

\( a \)

\textit{Named character}. Interpret bytes as named characters. Only the least significant seven bits of each byte will be used for this type specification. Bytes with the values listed in the following table will be written using the corresponding names for those characters.

\textbf{Named Characters in od}

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>Value</th>
<th>Name</th>
<th>Value</th>
<th>Name</th>
<th>Value</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>\000</td>
<td>nul</td>
<td>\001</td>
<td>scr</td>
<td>\002</td>
<td>stx</td>
<td>\003</td>
<td>etx</td>
</tr>
<tr>
<td>\004</td>
<td>eot</td>
<td>\005</td>
<td>enq</td>
<td>\006</td>
<td>ack</td>
<td>\007</td>
<td>bel</td>
</tr>
<tr>
<td>\010</td>
<td>bs</td>
<td>\011</td>
<td>ht</td>
<td>\012</td>
<td>lf</td>
<td>\013</td>
<td>vt</td>
</tr>
<tr>
<td>\014</td>
<td>ff</td>
<td>\015</td>
<td>cr</td>
<td>\016</td>
<td>so</td>
<td>\017</td>
<td>si</td>
</tr>
<tr>
<td>\020</td>
<td>dle</td>
<td>\021</td>
<td>dcl</td>
<td>\022</td>
<td>dc2</td>
<td>\023</td>
<td>dc3</td>
</tr>
<tr>
<td>\024</td>
<td>dc4</td>
<td>\025</td>
<td>nak</td>
<td>\026</td>
<td>syn</td>
<td>\027</td>
<td>etb</td>
</tr>
<tr>
<td>\030</td>
<td>can</td>
<td>\031</td>
<td>em</td>
<td>\032</td>
<td>sub</td>
<td>\033</td>
<td>esc</td>
</tr>
<tr>
<td>\034</td>
<td>fs</td>
<td>\035</td>
<td>gs</td>
<td>\036</td>
<td>rs</td>
<td>\037</td>
<td>us</td>
</tr>
</tbody>
</table>
Character. Interpret bytes as single-byte or multibyte characters specified by the current setting of the LC_CTYPE locale category. Printable multibyte characters are written in the area corresponding to the first byte of the character; the two character sequence ** is written in the area corresponding to each remaining byte in the character, as an indication that the character is continued. Certain non-graphic characters appear as C escapes: \0, \a, \b, \f, \n, \r, \t, \v. Other non-printable characters appear as one three-digit octal number for each byte in the character.

The type specification characters d, f, o, u, and x can be followed by an optional unsigned decimal integer that specifies the number of bytes to be transformed by each instance of the output type.

Floating point. Can be followed by an optional F, D, or L indicating that the conversion should be applied to an item of type float, double, or long double, respectively.

Signed decimal, octal, unsigned decimal, and hexadecimal, respectively. Can be followed by an optional C, S, I, or L indicating that the conversion should be applied to an item of type char, short, int, or long, respectively.

Multiple types can be concatenated within the same type_string and multiple -t options can be specified. Output lines are written for each type specified in the order in which the type specification characters are specified.

-\v
Show all input data (verbose). Without the -v option, all groups of output lines that would be identical to the immediately preceding output line (except for byte offsets), will be replaced with a line containing only an asterisk (*).

-\x
Interpret words in hex. This is equivalent to -t x2.

-\X
Interpret long words in hex. This is equivalent to -t x4.

** OPERANDS **
The following operands are supported for both /usr/bin/od and /usr/xpg4/bin/od:

- Use the standard input in addition to any files specified. When this operand is not given, the standard input is used only if no file
operands are specified.

The following operands are supported for /usr/bin/od only:

`file`

A path name of a file to be read. If no `file` operands are specified, the standard input will be used. If there are no more than two operands, none of the `-A`, `-j`, `-N`, or `-t` options is specified, and any of the following are true:

1. the first character of the last operand is a plus sign (+)
2. the first character of the second operand is numeric
3. the first character of the second operand is `x` and the second character of the second operand is a lower-case hexadecimal character or digit
4. the second operand is named "x"
5. the second operand is named "."

then the corresponding operand is assumed to be an offset operand rather than a file operand.

Without the `-N` count option, the display continues until an end-of-file is reached.

The following operands are supported for /usr/bin/od only:

`file`

Same as /usr/bin/od, except only one of the first two conditions must be true.

The following operands are supported for /usr/xpg4/bin/od only:

`file`

Same as /usr/bin/od, except only one of the first two conditions must be true.
Description of offset_string is the same as for /usr/bin/od.

ENVIRONMENT VARIABLES
See environ(5) for descriptions of the following environment variables that affect the execution of od: LC_CTYPE, LC_MESSAGES, LC_NUMERIC, and NLSPATH.

EXIT STATUS
The following exit values are returned:
0    Successful completion.
>0   An error occurred.

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

```
ATTRIBUTES

/usr/bin/od

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWtoo</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

/usr/xpg4/bin/od

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWxcu4</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>
```

SEE ALSO
sed(1), attributes(5), environ(5), XPG4(5)
on(1)

NAME
on – execute a command on a remote system, but with the local environment

SYNOPSIS
on [-i] [-d] [-n] host command [argument] ...

DESCRIPTION
The on program is used to execute commands on another system, in an environment
similar to that invoking the program. All environment variables are passed, and the
current working directory is preserved. To preserve the working directory, the
working file system must be either already mounted on the host or be exported to it.
Relative path names will only work if they are within the current file system; absolute
path names may cause problems.

The standard input is connected to the standard input of the remote command, and
the standard output and the standard error from the remote command are sent to the
 corresponds...
BUGS

When the working directory is remote mounted over NFS, a CTRL-Z hangs the window.

Root cannot use on.
**NAME**

optisa – determine which variant instruction set is optimal to use

**SYNOPSIS**

```
optisa instruction_set...
```

**DESCRIPTION**

optisa prints which `instruction_set` out of the ones specified in the command will perform best on this machine. In this case, “best” is defined by the order in which instruction set names are returned by `isalist(1)`. Possible values for `instruction_set` are given in `isalist(5)`.

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

**EXIT STATUS**

The following exit values are returned:

- 0 One of the `instruction_set` values you specified is printed by this command.
- 1 There is no output; that is, this machine cannot use any `instruction_set` that you specified with the `optisa` command.

**SEE ALSO**

`isalist(1), uname(1), attributes(5), isalist(5)`

**NOTES**

`optisa` is preferable to `uname -p` or `uname -m` (see `uname(1)`) in determining which of several binary versions of a given program should be used on the given machine.
### NAME
pack, pcat, unpack – compress and expand files

### SYNOPSIS
```
pack [-f] [+] file...
p_cat file...
unpack file...
```

### pack
The `pack` command attempts to store the specified files in a compressed form.
Wherever possible (and useful), each input file `file` is replaced by a packed file `file.z` with the same access modes, access and modified dates, and owner as those of `file`. If `pack` is successful, `file` will be removed.

The amount of compression obtained depends on the size of the input file and the character frequency distribution. Because a decoding tree forms the first part of each `.z` file, it is usually not worthwhile to pack files smaller than three blocks, unless the character frequency distribution is very skewed, which may occur with printer plots or pictures.

Typically, text files are reduced to 60-75% of their original size. Load modules, which use a larger character set and have a more uniform distribution of characters, show little compression, the packed versions being about 90% of the original size.

`pack` returns a value that is the number of files that it failed to compress.

No packing will occur if:
- the file appears to be already packed
- the file name has more than 14 – 2 bytes
- the file has links
- the file is a directory
- the file cannot be opened
- the file is empty
- no disk storage blocks will be saved by packing
- a file called `file.z` already exists
- the `.z` file cannot be created
- an I/O error occurred during processing.

The last segment of the file name must contain no more than 14 – 2 bytes to allow space for the appended `.z` extension. Directories cannot be compressed.

### pcat
The `pcat` command does for packed files what `cat(1)` does for ordinary files, except that `pcat` cannot be used as a filter. The specified files are unpacked and written to the standard output.

`pcat` returns the number of files it was unable to unpack. Failure may occur if:
- the file cannot be opened;
- the file does not appear to be the output of `pack`.

---

1070 man pages section 1: User Commands • Last Revised 20 Dec 1996
The `unpack` command expands files created by `pack`. For each file specified in the command, a search is made for a file called `file.z` (or just `file`, if `file` ends in `.z`). If this file appears to be a packed file, it is replaced by its expanded version. The new file has the `.z` suffix stripped from its name, and has the same access modes, access and modification dates, and owner as those of the packed file.

`unpack` returns a value that is the number of files it was unable to unpack. Failure may occur for the same reasons that it may in `pcat`, as well as for the following:

- a file with the “unpacked” name already exists;
- the unpacked file cannot be created.
- the filename (excluding the `.z` extension) has more than 14 bytes.

### OPTIONS

The following options are supported by `pack`:

- `-f` Forces packing of `file`. This is useful for causing an entire directory to be packed even if some of the files will not benefit. Packed files can be restored to their original form using `unpack` or `pcat`.

### OPERANDS

The following operands are supported:

- `file` A path name of a file to be packed, unpacked, or pcated; `file` can include or omit the `.z` suffix.

- `pack` uses Huffman (minimum redundancy) codes on a byte-by-byte basis. If the `–` argument is used, an internal flag is set that causes the number of times each byte is used, its relative frequency, and the code for the byte to be printed on the standard output. Additional occurrences of `–` in place of `file` will cause the internal flag to be set and reset.

### USAGE

See `largefile(5)` for the description of the behavior of `pack`, `pcat`, and `unpack` when encountering files greater than or equal to 2 Gbyte (`2^{31}` bytes).

### EXAMPLES

#### EXAMPLE 1 Viewing a Packed File

To view a packed file named `file.z` use:

```
example% pcat file.z
```

or just:

```
example% pcat file
```

#### EXAMPLE 2 Making and Unpacked Copy:

To make an unpacked copy, say `nnn`, of a packed file named `file.z` (without destroying `file.z`) use the command:

```
example% pcat file >nnn
```
ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of pack, pcat, and unpack: LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred. The number of files the command failed to pack/unpack is returned.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

SEE ALSO

cat(1), compress(1), zcat(1), attributes(5), environ(5), largefile(5)
NAME                 pagesize – display the size or sizes of a page of memory
SYNOPSIS            /usr/bin/pagesize [-a]
DESCRIPTION         The pagesize utility prints the default size of a page of memory in bytes, as returned by getpagesize(3C). This program is useful in constructing portable shell scripts.
OPTIONS             The following option is supported:
                      -a       Prints out all possible hardware address translation sizes supported by the system.
ATTRIBUTES           See attributes(5) for descriptions of the following attributes:
                      ATTRIBUTE TYPE | ATTRIBUTE VALUE
                      Availability   | SUNWcsu
SEE ALSO             getpagesize(3C), getpagesize(3C), attributes(5)
NAME  pargs – print process arguments, environment variables, or auxiliary vector

SYNOPSIS  pargs [-aceFx] [pid | core...]

DESCRIPTION  The pargs utility examines a target process or process core file and prints arguments, environment variables and values, or the process auxiliary vector.

pargs outputs unprintable characters as escaped octal in the format \xxx, unless the character is one of the characters specified in the “Escape Sequences” section of formats(5), in which case the character is printed as specified in that section.

pargs attempts to be sensitive to the locale of the target process. If the target process and the pargs process do not share a common character encoding, pargs attempts to employ the iconv(3C) facility to generate a printable version of the extracted strings. In the event that such a conversion is impossible, strings are displayed as 7-bit ASCII.

OPTIONS  The following options are supported:
- a  Prints process arguments as contained in argv[] (default).
- c  Treats strings in the target process as though they were encoded in 7-bit ASCII, regardless of the locale of the target. The use of iconv(3C) is suppressed.
- e  Prints process environment variables and values as pointed at by the _environ symbol or by pr_envp in /proc/pid/psinfo.
- F  Force. Grabs the target process even if another process has control.
- x  Prints process auxiliary vector.

OPERANDS  The following operands are supported:
pid  Process ID list.
core  Process core file.

EXIT STATUS  The following exit values are returned:
0  Successful operation.
non-zero  An error has occurred (such as no such process, permission denied, or invalid option).

FILES  /proc/pid/*  Process information and control files.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu (32-bit)</td>
</tr>
<tr>
<td></td>
<td>SUNWesxu (64-bit)</td>
</tr>
</tbody>
</table>
pargs(1)

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface Stability</td>
<td>Evolving</td>
</tr>
</tbody>
</table>

SEE ALSO proc(1), iconv(3C), proc(4), ascii(5), attributes(5), environ(5), formats(5)
The `passwd` command changes the password or lists password attributes associated with the user’s login name. Additionally, privileged users may use `passwd` to install or change passwords and attributes associated with any login name.

When used to change a password, `passwd` prompts everyone for their old password, if any. It then prompts for the new password twice. When the old password is entered, `passwd` checks to see if it has “aged” sufficiently. If “aging” is insufficient, `passwd` terminates; see `pwconv(1M)`, `nistbladm(1)`, and `shadow(4)` for additional information.

When LDAP, NIS, or NIS+ is in effect on a system, `passwd` changes the NIS or NIS+ database. The NIS or NIS+ password may be different from the password on the local machine. If NIS or NIS+ is running, use `passwd -r` to change password information on the local machine.

The `pwconv` command creates and updates `/etc/shadow` with information from `/etc/passwd`. `pwconv` relies on a special value of ‘x’ in the password field of `/etc/passwd`. This value of ‘x’ indicates that the password for the user is already in `/etc/shadow` and should not be modified.

If aging is sufficient, a check is made to ensure that the new password meets construction requirements. When the new password is entered a second time, the two copies of the new password are compared. If the two copies are not identical, the cycle of prompting for the new password is repeated for, at most, two more times.

Passwords must be constructed to meet the following requirements:

- Each password must have `PASSLENGTH` characters, where `PASSLENGTH` is defined in `/etc/default/passwd` and is set to 6. Only the first eight characters are significant.
Each password must contain at least two alphabetic characters and at least one numeric or special character. In this case, "alphabetic" refers to all upper or lower case letters.

Each password must differ from the user’s login name and any reverse or circular shift of that login name. For comparison purposes, an upper case letter and its corresponding lower case letter are equivalent.

New passwords must differ from the old by at least three characters. For comparison purposes, an upper case letter and its corresponding lower case letter are equivalent.

If all requirements are met, by default, the passwd command will consult /etc/nsswitch.conf to determine in which repositories to perform password update. It searches the passwd and passwd_compat entries. The sources (repositories) associated with these entries will be updated. However, the password update configurations supported are limited to the following cases. Failure to comply with the configurations will prevent users from logging onto the system. The password update configurations are:

- passwd: files
- passwd: files ldap
- passwd: files nis
- passwd: files nisplus
- passwd: compat (==> files nis)
- passwd: compat (==> files ldap)
  - passwd_compat: ldap
- passwd: compat (==> files nisplus)
  - passwd_compat: nisplus

Network administrators, who own the NIS+ password table, may change any password attributes.

In the files case, super-users (for instance, real and effective uid equal to 0, see id(1M) and su(1M)) may change any password. Hence, passwd does not prompt privileged users for the old password. Privileged users are not forced to comply with password aging and password construction requirements. A privileged user can create a null password by entering a carriage return in response to the prompt for a new password. (This differs from passwd -d because the "password" prompt will still be displayed.) If NIS is in effect, superuser on the root master can change any password without being prompted for the old NIS passwd, and is not forced to comply with password construction requirements.

Normally, passwd entered with no arguments will change the password of the current user. When a user logs in and then invokes su(1M) to become super-user or another user, passwd will change the original user’s password, not the password of the super-user or the new user.
Any user may use the `-s` option to show password attributes for his or her own login name, provided they are using the `-r nisplus` argument. Otherwise, the `-s` argument is restricted to the superuser.

The format of the display will be:

```
name status mm/dd/yy min max warn
```

or, if password aging information is not present,

```
name status
```

where

- **name**: The login ID of the user.
- **status**: The password status of name: PS stands for passworded or locked, LK stands for locked, and NP stands for no password.
- **mm/dd/yy**: The date password was last changed for name. Notice that all password aging dates are determined using Greenwich Mean Time (Universal Time) and therefore may differ by as much as a day in other time zones.
- **min**: The minimum number of days required between password changes for name. MINWEEKS is found in `/etc/default/passwd` and is set to NULL.
- **max**: The maximum number of days the password is valid for name. MAXWEEKS is found in `/etc/default/passwd` and is set to NULL.
- **warn**: The number of days relative to max before the password expires and the name will be warned.

**Security**

`passwd` uses pam(3PAM) for password management. The PAM configuration policy, listed through `/etc/pam.conf`, specifies the password modules to be used for `passwd`. Here is a partial `pam.conf` file with entries for the `passwd` command using the `passwd-auth` module:

```
passwd auth required pam_passwd_auth.so.1
```

If there are no entries for the `passwd` service, then the entries for the "other" service will be used. If multiple password modules are listed, then the user may be prompted for multiple passwords.

**OPTIONS**

The following options are supported:

- `-a` Shows password attributes for all entries. Use only with the `-s` option. name must not be provided. For the nisplus repository, this will show only the entries in the NIS+ password table in the local domain that the invoker is authorized to "read". For the files repository, this is restricted to the superuser.
passwd(1)

**-D domainname**  Consults the passwd.org_dir table in domainname. If this option is not specified, the default domainname returned by nis_local_directory(3NSL) will be used. This domain name is the same as that returned by domainname(1M).

**-e**  Changes the login shell. For the files repository, this only works for the super-user. Normal users may change the ldap, nis, or nisplus repositories. The choice of shell is limited by the requirements of getusershell(3C). If the user currently has a shell that is not allowed by getusershell, only root may change it.

**-g**  Changes the gecos (finger) information. For the files repository, this only works for the superuser. Normal users may change the ldap, nis, or nisplus repositories.

**-h**  Changes the home directory.

**-r**  Specifies the repository to which an operation is applied. The supported repositories are files, ldap, nis, or nisplus.

**-s name**  Shows password attributes for the login name. For the nisplus repository, this works for everyone. However for the files repository, this only works for the superuser. It does not work at all for the nis repository which does not support password aging.

**Privileged User Options**

Only a privileged user can use the following options:

**-d**  Deletes password for name. The login name will not be prompted for password. It is only applicable to the files repository.

**-f**  Forces the user to change password at the next login by expiring the password for name.

**-l**  Locks password entry for name.

**-n min**  Sets minimum field for name. The min field contains the minimum number of days between password changes for name. If min is greater than max, the user may not change the password. Always use this option with the -x option, unless max is set to -1 (aging turned off). In that case, min need not be set.

**-w warn**  Sets warn field for name. The warn field contains the number of days before the password expires and the user is warned. This option is not valid if password aging is disabled.

**-x max**  Sets maximum field for name. The max field contains the number of days that the password is valid for name. The aging for name will be turned off immediately if max is set to -1. If it is set to 0, then the user is forced to change the password at the next login session and aging is turned off.

**OPERANDS** The following operand is supported:
passwd(1)

**name**  
User login name.

**ENVIRONMENT VARIABLES**

If any of the LC_* variables, that is, LC_CTYPE, LC_MESSAGES, LC_TIME, LC_COLLATE, LC_NUMERIC, and LC_MONETARY (see environ(5)), are not set in the environment, the operational behavior of passwd for each corresponding locale category is determined by the value of the LANG environment variable. If LC_ALL is set, its contents are used to override both the LANG and the other LC_* variables. If none of the above variables is set in the environment, the "C" (U.S. style) locale determines how passwd behaves.

**LC_CTYPE**  
Determines how passwd handles characters. When LC_CTYPE is set to a valid value, passwd can display and handle text and filenames containing valid characters for that locale. passwd can display and handle Extended Unix Code (EUC) characters where any individual character can be 1, 2, or 3 bytes wide. passwd can also handle EUC characters of 1, 2, or more column widths. In the "C" locale, only characters from ISO 8859-1 are valid.

**LC_MESSAGES**  
Determines how diagnostic and informative messages are presented. This includes the language and style of the messages, and the correct form of affirmative and negative responses. In the "C" locale, the messages are presented in the default form found in the program itself (in most cases, U.S. English).

**EXIT STATUS**

The passwd command exits with one of the following values:

0  
Success.

1  
Permission denied.

2  
Invalid combination of options.

3  
Unexpected failure. Password file unchanged.

4  
Unexpected failure. Password file(s) missing.

5  
Password file(s) busy. Try again later.

6  
Invalid argument to option.

7  
Aging option is disabled.

8  
No memory.

9  
System error.

10  
Account expired.

**FILES**

/etc/oshadow  
Password file.

/etc/shells  
Shadow password file.
Default values can be set for the following flags in /etc/default/passwd. For example: MAXWEEKS=26

<table>
<thead>
<tr>
<th>ATTRIBUTE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAXWEEKS</td>
<td>Maximum time period that password is valid.</td>
</tr>
<tr>
<td>MINWEEKS</td>
<td>Minimum time period before the password can be changed.</td>
</tr>
<tr>
<td>PASSLENGTH</td>
<td>Minimum length of password, in characters.</td>
</tr>
<tr>
<td>WARNWEEKS</td>
<td>Time period until warning of date of password’s ensuing expiration.</td>
</tr>
</tbody>
</table>

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

SEE ALSO

finger(1), login(1), nistbladm(1), domainname(1M), eeprom(1M), id(1M), passmgmt(1M), pwconv(1M), su(1M), useradd(1M), userdel(1M), usermod(1M), crypt(3C), getpwnam(3C), getspnam(3C), getusershell(3C), nis_local_directory(3NSL), pam(3PAM), loginlog(4), nsswitch.conf(4), pam.conf(4), passwd(4), shadow(4), attributes(5), environ(5), pam_authhtok_check(5), pam_authhtok_get(5), pam_authhtok_store(5), pam_dhkeys(5), pam_ldap(5), pam_unix(5), pam_unix_account(5), pam_unix_auth(5), pam_unix_session(5)

NOTES

The pam_unix(5) module might not be supported in a future release. Similar functionality is provided by pam_unix_account(5), pam_unix_auth(5), pam_unix_session(5), pam_authhtok_check(5), pam_authhtok_get(5), pam_authhtok_store(5), pam_dhkeys(5), and pam_passwd_auth(5).

The nispasswd and yp passwd commands are wrappers around passwd. Use of nispasswd and yp passwd is discouraged. Use passwd -r repository_name instead.

NIS+ might not be supported in future releases of the Solaris™ Operating Environment. Tools to aid the migration from NIS+ to LDAP are available in the Solaris 9 operating environment. For more information, visit http://www.sun.com/directory/nisplus/transition.html.
paste(1)

NAME    paste – merge corresponding or subsequent lines of files

SYNOPSIS  paste [-s] [-a list] file...

DESCRIPTION  The paste utility will concatenate the corresponding lines of the given input files, and write the resulting lines to standard output.

The default operation of paste will concatenate the corresponding lines of the input files. The NEWLINE character of every line except the line from the last input file will be replaced with a TAB character.

If an EOF (end-of-file) condition is detected on one or more input files, but not all input files, paste will behave as though empty lines were read from the files on which EOF was detected, unless the -s option is specified.

OPTIONS  The following options are supported:

- a list  Unless a backslash character (\) appears in list, each character in list is an element specifying a delimiter character. If a backslash character appears in list, the backslash character and one or more characters following it are an element specifying a delimiter character as described below. These elements specify one or more delimiters to use, instead of the default TAB character, to replace the NEWLINE character of the input lines. The elements in list are used circularly; that is, when the list is exhausted the first element from the list is reused.

When the -s option is specified:

- The last newline character in a file will not be modified.
- The delimiter will be reset to the first element of list after each file operand is processed.

When the option is not specified:

- The NEWLINE characters in the file specified by the last file will not be modified.
- The delimiter will be reset to the first element of list each time a line is processed from each file.

If a backslash character appears in list, it and the character following it will be used to represent the following delimiter characters:

\n            Newline character.
\t            Tab character.
\\           Backslash character.
\0            Empty string (not a null character). If \0 is immediately followed by the character x, the character X, or any
character defined by the LC_CTYPE digit keyword, the results are unspecified.

If any other characters follow the backslash, the results are unspecified.

-s Concatenate all of the lines of each separate input file in command line order. The NEWLINE character of every line except the last line in each input file will be replaced with the TAB character, unless otherwise specified by the -d option.

OPERANDS
The following operand is supported:

file A path name of an input file. If - is specified for one or more of the files, the standard input will be used; the standard input will be read one line at a time, circularly, for each instance of -. Implementations support pasting of at least 12 file operands.

USAGE
See largefile(5) for the description of the behavior of paste when encountering files greater than or equal to 2 Gbyte (2^31 bytes).

EXAMPLES

EXAMPLE 1 List a directory in one column.

ls | paste -d " "

EXAMPLE 2 List a directory in four columns.

ls | paste - - - -

EXAMPLE 3 Combine pairs of lines from a file into single lines.

paste -s -d"\t\n" file

ENVIRONMENT VARIABLES
See environ(5) for descriptions of the following environment variables that affect the execution of paste: LC_CTYPE and LC_MESSAGES.

EXIT STATUS
The following exit values are returned:

0 Successful completion.

>0 An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>
paste(1)

SEE ALSO  cut(1), grep(1), pr(1), attributes(5), environ(5), largefile(5)

DIAGNOSTICS

"line too long"       Output lines are restricted to 511 characters.
"too many files"      Except for -s option, no more than 12 input files may
                      be specified.
"no delimiters"       The -d option was specified with an empty list.
"cannot open file"    The specified file cannot be opened.
The `patch` command reads a source (patch) file containing any of the three forms of difference (diff) listings produced by the `diff(1)` command (normal, context or in the style of `ed(1)`) and apply those differences to a file. By default, `patch` reads from the standard input.

If the patch file contains more than one patch, `patch` will attempt to apply each of them as if they came from separate patch files. (In this case the name of the patch file must be determinable for each diff listing.)

The following options are supported:

- `-b` Saves a copy of the original contents of each modified file, before the differences are applied, in a file of the same name with the suffix `.orig` appended to it. If the file already exists, it will be overwritten. If multiple patches are applied to the same file, the `.orig` file will be written only for the first patch. When the `-o` outfile option is also specified, file `.orig` will not be created but, if outfile already exists, outfile `.orig` will be created.

- `-c` Interprets the patch file as a context difference (the output of the command `diff` when the `-c` or `-C` options are specified).

- `-d dir` Changes the current directory to `dir` before processing as described in EXTENDED DESCRIPTION.

- `-D define` Marks changes with the C preprocessor construct:

```
#define define
...
#endif
```

The option-argument `define` will be used as the differentiating symbol.

- `-e` Interprets the patch file as an `ed` script, rather than a `diff` script.

- `-i patchfile` Reads the patch information from the file named by the path name `patchfile`, rather than the standard input.

- `-l` (The letter ell.) Causes any sequence of blank characters in the difference script to match any sequence of blank characters in the input file. Other characters will be matched exactly.

- `-n` Interprets the script as a normal difference.

- `-N` Ignores patches where the differences have already been applied to the file; by default, already-applied patches are rejected.
Instead of modifying the files (specified by the file operand or the difference listings) directly, writes a copy of the file referenced by each patch, with the appropriate differences applied, to outfile. Multiple patches for a single file will be applied to the intermediate versions of the file created by any previous patches, and will result in multiple, concatenated versions of the file being written to outfile.

For all path names in the patch file that indicate the names of files to be patched, deletes num path name components from the beginning of each path name. If the path name in the patch file is absolute, any leading slashes are considered the first component (that is, -p 1 removes the leading slashes). Specifying -p 0 causes the full path name to be used. If -p is not specified, only the basename (the final path name component) is used.

Reverses the sense of the patch script. That is, assumes that the difference script was created from the new version to the old version. The -R option cannot be used with ed scripts. patch attempts to reverse each portion of the script before applying it. Rejected differences will be saved in swapped format. If this option is not specified, and until a portion of the patch file is successfully applied, patch attempts to apply each portion in its reversed sense as well as in its normal sense. If the attempt is successful, the user will be prompted to determine if the -R option should be set.

Overrides the default reject file name. In the default case, the reject file will have the same name as the output file, with the suffix .rej appended to it. See Patch Application.

Interprets the patch file as a unified context difference, that is, the output of the command diff when the -u or -U options are specified.

The following operand is supported:

file A path name of a file to patch.

The -R option will not work with ed scripts because there is too little information to reconstruct the reverse operation.

The -p option makes it possible to customize a patch file to local user directory structures without manually editing the patch file. For example, if the file name in the patch file was /curds/whey/src/blurfl/blurfl.c:

- Setting -p 0 gives the entire path name unmodified.
- Setting -p 1 gives:

  curds/whey/src/blurfl/blurfl.c
Without the leading slash, -p 4 gives:

`blurfl/blurfl.c`

Not specifying -p at all gives:

`blurfl.c`

When using -b in some file system implementations, the saving of a .orig file may produce unwanted results. In the case of 12-, 13-, or 14-character file names, on file systems supporting 14-character maximum file names, the .orig file will overwrite the new file.

**ENVIRONMENT VARIABLES**

See environ(5) for descriptions of the following environment variables that affect the execution of patch: LC_CTYPE, LC_MESSAGES, LC_TIME, and NLSPATH.

**OUTPUT FILES**

The output of patch the save files (.orig suffixes) and the reject files (.rej suffixes) will be text files.

**EXTENDED DESCRIPTION**

A patch file may contain patching instructions for more than one file. File names are determined as specified in Patch Determination. When the -b option is specified, for each patched file, the original will be saved in a file of the same name with the suffix .orig appended to it.

For each patched file, a reject file may also be created as noted in Patch Application. In the absence of an -r option, the name of this file will be formed by appending the suffix .rej to the original file name.

**Patch File Format**

The patch file must contain zero or more lines of header information followed by one or more patches. Each patch must contain zero or more lines of file name identification in the format produced by diff -c, and one or more sets of diff output, which are customarily called hunks.

patch recognizes the following expression in the header information:

```
Index: pathname
```

The file to be patched is named `pathname`.

If all lines (including headers) within a patch begin with the same leading sequence of blank characters, patch will remove this sequence before proceeding. Within each patch, if the type of difference is context, patch recognizes the following expressions:

```
* * * filename timestamp
```

The patches arose from `filename`.

```
−−− filename timestamp
```

The patches should be applied to `filename`.

Each hunk within a patch must be the diff output to change a line range within the original file. The line numbers for successive hunks within a patch must occur in ascending order.
If no file operand is specified, **patch** performs the following steps to obtain a path name:

1. If the patch contains the strings *** and −−−, **patch** strips components from the beginning of each path name (depending on the presence or value of the -p option), then tests for the existence of both files in the current directory (or directory specified with the -d option).

2. If both files exist, **patch** assumes that no path name can be obtained from this step. If the header information contains a line with the string Index:, **patch** strips components from the beginning of the path name (depending on -p), then tests for the existence of this file in the current directory (or directory specified with the -d option).

3. If an SCCS directory exists in the current directory, **patch** will attempt to perform a get -e SCCS/s_filename command to retrieve an editable version of the file.

4. If no path name can be obtained by applying the previous steps, or if the path names obtained do not exist, **patch** will write a prompt to standard output and request a file name interactively from standard input.

If the -c, -e, -n, or -u option is present, **patch** will interpret information within each hunk as a context difference, an ed difference, a normal difference, or a unified context difference, respectively. In the absence of any of these options, **patch** determines the type of difference based on the format of information within the hunk.

For each hunk, **patch** begins to search for the place to apply the patch at the line number at the beginning of the hunk, plus or minus any offset used in applying the previous hunk. If lines matching the hunk context are not found, **patch** scans both forwards and backwards at least 1000 bytes for a set of lines that match the hunk context.

If no such place is found and it is a context difference, then another scan will take place, ignoring the first and last line of context. If that fails, the first two and last two lines of context will be ignored and another scan will be made. Implementations may search more extensively for installation locations.

If no location can be found, **patch** will append the hunk to the reject file. The rejected hunk will be written in context-difference format regardless of the format of the patch file. If the input was a normal or ed-style difference, the reject file may contain differences with zero lines of context. The line numbers on the hunks in the reject file may be different from the line numbers in the patch file since they will reflect the approximate locations for the failed hunks in the new file rather than the old one.

If the type of patch is an ed diff, the implementation may accomplish the patching by invoking the ed command.

**EXIT STATUS**

The following exit values are returned:

<table>
<thead>
<tr>
<th>Exit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>1</td>
<td>One or more lines were written to a reject file.</td>
</tr>
</tbody>
</table>
>1 An error occurred.

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

**SEE ALSO**
ed(1), diff(1), attributes(5), environ(5)
pathchk(1)

NAME  pathchk – check path names

SYNOPSIS  pathchk [-p] path...

DESCRIPTION The pathchk command will check that one or more path names are valid (that is, they could be used to access or create a file without causing syntax errors) and portable (that is, no filename truncation will result). More extensive portability checks are provided by the -p option.

By default, pathchk will check each component of each path operand based on the underlying file system. A diagnostic will be written for each path operand that:

- is longer than PATH_MAX bytes.
- contains any component longer than NAME_MAX bytes in its containing directory
- contains any component in a directory that is not searchable
- contains any character in any component that is not valid in its containing directory.

The format of the diagnostic message is not specified, but will indicate the error detected and the corresponding path operand.

It will not be considered an error if one or more components of a path operand do not exist as long as a file matching the path name specified by the missing components could be created that does not violate any of the checks specified above.

OPTIONS The following option is supported:

- -p Instead of performing checks based on the underlying file system, write a diagnostic for each path operand that:
  - is longer than _POSIX_PATH_MAX bytes
  - contains any component longer than _POSIX_NAME_MAX bytes
  - contains any character in any component that is not in the portable filename character set.

OPERANDS The following operand is supported:

path A path to be checked.

USAGE See largefile(5) for the description of the behavior of pathchk when encountering files greater than or equal to 2 Gbyte (2^31 bytes).

EXAMPLES EXAMPLE 1 Examples of the pathchk command.

To verify that all paths in an imported data interchange archive are legitimate and unambiguous on the current system:

```
pax -f archive | sed -e '/ == .*/s///' | xargs pathchk
if [ $? -eq 0 ]
then
  pax -r -f archive
else
```

1090 man pages section 1: User Commands • Last Revised 1 Feb 1995
EXAMPLE 1 Examples of the pathchk command.  (Continued)

    echo Investigate problems before importing files.
    exit 1
fi

To verify that all files in the current directory hierarchy could be moved to any system conforming to the X/Open specification that also supports the $pax(1)$ command:

    find . -print | xargs pathchk -p
    if [ $? -eq 0 ]
      then
        pax -w -f archive .
      else
        echo Portable archive cannot be created.
        exit 1
    fi

To verify that a user-supplied path names a readable file and that the application can create a file extending the given path without truncation and without overwriting any existing file:

    case $- in
      *C*) reset="";;
      *)     reset="set +C"
            set -C;;
    esac
    test -r "$path" && pathchk "$path.out" &&
    rm "$path.out" > "$path.out"
    if [ $? -ne 0 ]; then
      printf "%%s: %%s not found or %%s.out fails \ncreation checks.\n" $0 "$path" "$path"
      $reset    # reset the noclobber option in case a trap
                  # on EXIT depends on it
      exit 1
    fi
    $reset
    PROCESSING < "$path" > "$path.out"

The following assumptions are made in this example:

1. PROCESSING represents the code that will be used by the application to use $path
   once it is verified that $path.out will work as intended.

2. The state of the noclobber option is unknown when this code is invoked and
   should be set on exit to the state it was in when this code was invoked. (The $reset
   variable is used in this example to restore the initial state.)

3. Note the usage of:

   rm "$path.out" > "$path.out"

   a. The pathchk command has already verified, at this point, that
      $path.out will not be truncated.

   b. With the noclobber option set, the shell will verify that $path.out
      does not already exist before invoking $rm.
EXAMPLE 1 Examples of the pathchk command.  (Continued)

c. If the shell succeeded in creating $path.out, rm will remove it so that
the application can create the file again in the PROCESSING step.
d. If the PROCESSING step wants the file to exist already when it is
invoked, the:

    rm "$path.out" > "$path.out"

should be replaced with:

    > "$path.out"

which will verify that the file did not already exist, but leave
$path.out in place for use by PROCESSING.

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the
execution of pathchk: LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS

The following exit values are returned:

0    All path operands passed all of the checks.
>0   An error occurred.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO

pax(1), test(1), attributes(5), environ(5), largefile(5)
**NAME**
pathconv – search FMLI criteria for filename

**SYNOPSIS**
```
pathconv [-f] [-v alias]
pathconv [-t] [-l] [-n num] [-v string]
```

**DESCRIPTION**
The pathconv function converts an alias to its pathname. By default, it takes the alias as a string from the standard input.

**OPTIONS**
- **-f**
  If -f is specified, the full path will be returned (this is the default).
- **-t**
  If -t is specified, pathconv will truncate a pathname specified in `string` in a format suitable for display as a frame title. This format is a shortened version of the full pathname, created by deleting components of the path from the middle of the string until it is under DISPLAY: 6 characters in length, and then inserting ellipses ( . . . ) between the remaining pieces. Ellipses are also used to show truncation at the ends of the strings if necessary, unless the -l option is given.
- **-l**
  If -l is specified, < and > will be used instead of ellipses ( . . . ) to indicate truncation at the ends of the string generated by the -t option. Using -l allows display of the longest possible string while still notifying users it has been truncated.
- **-n num**
  If -n is specified, num is the maximum length of the string (in characters) generated by the -t option. The argument num can be any integer from 1 to 255.
- **-valias | string**
  If the -v option is used, then alias or string can be specified when pathconv is called. The argument alias must be an alias defined in the `alias_file` named when fmli was invoked. The argument string can only be used with the -t option and must be a pathname.

**EXAMPLES**

### EXAMPLE 1
A sample that uses `pathconv` to construct the menu title. It searches for MYPATH in the `alias_file` named when fmli command.

Here is a menu descriptor that uses `pathconv` to construct the menu title. It searches for MYPATH in the `alias_file` named when fmli was invoked:

```bash
menu='pathconv -v MYPATH/ls'
```

where there is a line in `alias_file` that defines MYPATH. For example, MYPATH=$HOME/bin:/usr/bin.

Here is a menu descriptor that takes alias from the standard input.

```bash
menu='echo MYPATH/ls | pathconv'
```
EXAMPLE 1 A sample that uses pathconv to construct the menu title. It searches for MYPATH in the alias file named when fmli command. (Continued)

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO fmli(1), attributes(5)
pax(1)

NAME
pax – portable archive interchange

SYNOPSIS
pax [-cdnv] [-f archive] [-s replstr] ... [pattern...]
pax -r [-diknuv] [-f archive] [-o options...] [-p string] ... [-s replstr] ... [pattern...]
pax -w [-dituvX@] [-b blocksize] [-a] [-f archive] [-o options] ... [-s replstr] ... [-x format] [file...]
pax -r -w [-diklntuvX@] [-p string] ... [-s replstr] ... [file...] directory

DESCRIPTION
The pax command reads, writes and writes lists of the members of archive files and
copy directory hierarchies. A variety of archive formats are supported. See the -x
format option.

Modes of Operations
The action to be taken depends on the presence of the -r and -w options. The four
combinations of -r and -w are referred to as the four modes of operation: list, read,
write, and copy modes, corresponding respectively to the four forms shown in the
SYNOPSIS.

list  In list mode (when neither -r nor -w are specified), pax writes the names
      of the members of the archive file read from the standard input, with path
      names matching the specified patterns, to standard output. If a named file
      has extended attributes, the extended attributes are also listed. If a named
      file is of type directory, the file hierarchy rooted at that file will be written
      out as well.

read  In read mode (when -r is specified, but -w is not), pax extracts the
      members of the archive file read from the standard input, with path names
      matching the specified patterns. If an extracted file is of type directory, the
      file hierarchy rooted at that file will be extracted as well. The extracted files
      is created relative to the current file hierarchy.

      The ownership, access and modification times, and file mode of the
      restored files are discussed under the -p option.

write In write mode (when -w is specified, but -r is not), pax writes the contents
      of the file operands to the standard output in an archive format. If no file
      operands are specified, a list of files to copy, one per line, will be read from
      the standard input. A file of type directory will include all of the files in the
      file hierarchy rooted at the file.

copy In copy mode (when both -r and -w are specified), pax copies the file
      operands to the destination directory.

      If no file operands are specified, a list of files to copy, one per line, will be
      read from the standard input. A file of type directory will include all of the
      files in the file hierarchy rooted at the file.
pax(1)

The effect of the copy is as if the copied files were written to an archive file and then subsequently extracted, except that there may be hard links between the original and the copied files. If the destination directory is a subdirectory of one of the files to be copied, the results are unspecified. It is an error if directory does not exist, is not writable by the user, or is not a directory.

In read or copy modes, if intermediate directories are necessary to extract an archive member, pax will perform actions equivalent to the mkdir(2) function, called with the following arguments:

- the intermediate directory used as the path argument
- the octal value of 777 or rwx (read, write, and execute permissions) as the mode argument (see chmod(1)).

If any specified pattern or file operands are not matched by at least one file or archive member, pax will write a diagnostic message to standard error for each one that did not match and exit with a non-zero exit status.

The supported archive formats are automatically detected on input. The default output archive format is tar(1).

If the selected archive format supports the specification of linked files, it is an error if these files cannot be linked when the archive is extracted. Any of the various names in the archive that represent a file can be used to select the file for extraction.

OPTIONS

The following options are supported:

- `-r` Reads an archive file from standard input.
- `-w` Writes files to the standard output in the specified archive format.
- `-a` Appends files to the end of the archive. This option will not work for some archive devices, such as 1/4-inch streaming tapes and 8mm tapes.
- `-b blocksize` Blocks the output at a positive decimal integer number of bytes per write to the archive file. Devices and archive formats may impose restrictions on blocking. Blocking is automatically determined on input. Portable applications must not specify a blocksize value larger than 32256. Default blocking when creating archives depends on the archive format. See the -x option below.
- `-c` Matches all file or archive members except those specified by the pattern or file operands.
- `-d` Causes files of type directory being copied or archived or archive members of type directory being extracted to match only the file or archive member itself and not the file hierarchy rooted at the file.
-f archive
   Specifies the path name of the input or output archive, overriding
   the default standard input (in list or read modes) or standard
   output (write mode).

-i
   Interactively renames files or archive members. For each archive
   member matching a pattern operand or file matching a file operand,
   a prompt will be written to the file /dev/tty. The prompt will
   contain the name of the file or archive member. A line will then be
   read from /dev/tty. If this line is blank, the file or archive
   member will be skipped. If this line consists of a single period, the
   file or archive member will be processed with no modification to
   its name. Otherwise, its name will be replaced with the contents of
   the line. The pax command will immediately exit with a non-zero
   exit status if end-of-file is encountered when reading a response or
   if /dev/tty cannot be opened for reading and writing.

-k
   Prevents the overwriting of existing files.

-l
   Links files. In copy mode, hard links will be made between the
   source and destination file hierarchies whenever possible.

-n
   Selects the first archive member that matches each pattern operand.
   No more than one archive member will be matched for each
   pattern, although members of type directory will still match the file
   hierarchy rooted at that file.

-o options
   Reserved for special format-specific options.

-p string
   Specifies one or more file characteristic options (privileges). The
   string operand must be a string specifying file
   characteristics to be retained or discarded on extraction. The string
   consists of the specification characters a, e, m, o, and p. Multiple
   characteristics can be concatenated within the same string and
   multiple -p options can be specified. The meaning of the
   specification characters are as follows:
   a  Does not preserve file access times.
   e  Preserves the user ID, group ID, file mode bits, access
      time, and modification time.
   m  Does not preserve file modification times.
   o  Preserves the user ID and group ID.
   p  Preserves the file mode bits. Other,
      implementation-dependent file-mode attributes may be
      preserved.
In the preceding list, “preserve” indicates that an attribute stored in the archive will be given to the extracted file, subject to the permissions of the invoking process. Otherwise, the attribute will be determined as part of the normal file creation action.

If neither the e nor the o specification character is specified, or the user ID and group ID are not preserved for any reason, pax will not set the setuid and setgid bits of the file mode.

If the preservation of any of these items fails for any reason, pax will write a diagnostic message to standard error. Failure to preserve these items will affect the final exit status, but will not cause the extracted file to be deleted.

If file-characteristic letters in any of the string option-arguments are duplicated or conflict with each other, the ones given last will take precedence. For example, if -p eme is specified, file modification times will be preserved.

-s replstr

Modifies file or archive member names named by pattern or file operands according to the substitution expression replstr, which is based on the ed(1) s (substitution) command, using the regular expression syntax on the regex(5) manual page. The concepts of “address” and “line” are meaningless in the context of the pax command, and must not be supplied. The format is:

-s / old/new/ [gp]

where, as in ed, old is a basic regular expression and new can contain an ampersand (&) or a \n backreference, where n is a digit. The old string also is permitted to contain newline characters.

Any non-null character can be used as a delimiter ( / shown here). Multiple -s expressions can be specified; the expressions will be applied in the order specified, terminating with the first successful substitution. The optional trailing g is as defined in the ed command. The optional trailing p causes successful substitutions to be written to standard error. File or archive member names that substitute to the empty string are ignored when reading and writing archives.

-t

Causes the access times of the archived files to be the same as they were before being read by pax.

-u

Ignores files that are older (having a less recent file modification time) than a pre-existing file or archive member with the same name.
read mode  An archive member with the same name as a file in the file system will be extracted if the archive member is newer than the file.

write mode  An archive file member with the same name as a file in the file system will be superseded if the file is newer than the archive member.

copy mode  The file in the destination hierarchy will be replaced by the file in the source hierarchy or by a link to the file in the source hierarchy if the file in the source hierarchy is newer.

-v  In list mode, produces a verbose table of contents (see Standard Output). Otherwise, writes archive member path names and extended attributes to standard error (see Standard Error).

-x format  Specifies the output archive format. The pax command recognizes the following formats:

cpio  The extended cpio(1) interchange format. See the IEEE 1003.1(1990) specifications. The default blocksize for this format for character special archive files is 5120. Implementations support all blocksize values less than or equal to 32256 that are multiples of 512.

This archive format allows files with UIDs and GIDs up to 262143 to be stored in the archive. Files with UIDs and GIDs greater than this value will be archived with the UID and GID of 60001.

ustar  The extended tar(1) interchange format. See the IEEE 1003.1(1990) specifications. The default blocksize for this format for character special archive files is 10240. Implementations support all blocksize values less than or equal to 32256 that are multiples of 512.

Any attempt to append to an archive file in a format different from the existing archive format will cause pax to exit immediately with a non-zero exit status.

This archive format allows files with UIDs and GIDs up to 2097151 to be stored in the archive. Files with UIDs and GIDs greater than this value will be archived with the UID and GID of 60001.
xustar

Similar to ustar. Will also allow archiving and extracting files whose size is greater than 8GB; whose UID, GID, devmajor, or devminor values are greater than 2097151; whose path (including filename) is greater than 255 characters; or whose linkname is greater than 100 characters. This option should not be used if the archive is to be extracted by an archiver that cannot handle the larger values.

-X

When traversing the file hierarchy specified by a path name, pax will not descend into directories that have a different device ID (st_dev, see stat(2)).

-@

When traversing the file hierarchy specified by a path name, pax will descend into the attribute directory for any file with extended attributes. Extended attributes go into the archive as special files. When this flag is used during file extraction, any extended attributes associated with a file being extracted are also extracted. Extended attribute files can only be extracted from an archive as part of a normal file extract. Attempts to explicitly extract attribute records are ignored.

The options that operate on the names of files or archive members (-c, -i, -n, -s, -u and -v) interact as follows. In read mode, the archive members are selected based on the user-specified pattern operands as modified by the -c, -n and -u options. Then, any -s and -i options will modify, in that order, the names of the selected files. The -v option will write names resulting from these modifications.

In write mode, the files are selected based on the user-specified path names as modified by the -n and -u options. Then, any -s and -i options will, in that order, modify the names of these selected files. The -v option will write names resulting from these modifications.

If both the -u and -n options are specified, pax does not consider a file selected unless it is newer than the file to which it is compared.

OPERANDS

The following operands are supported:

directory The destination directory path name for copy mode.

file A path name of a file to be copied or archived.

pattern A pattern matching one or more path names of archive members. A pattern must conform to the pattern matching notation found on the fnmatch(5) manual page. The default, if no pattern is specified, is to select all members in the archive.

Standard Output

In write mode, if -f is not specified, the standard output will be the archive formatted according to cpio or ustar. (See -x format.)
In list mode, the table of contents of the selected archive members will be written to standard output using the following format:

```
%s
```

If the -v option is specified in list mode, the table of contents of the selected archive members will be written to standard output using the following formats:

For path names representing hard links to previous members of the archive:

```
%s==%s
```

<ls -l listing>, linkname

For all other path names:

```
%s
```

where <ls -l listing> is the format specified by the ls command with the -l option. When writing path names in this format, it is unspecified what is written for fields for which the underlying archive format does not have the correct information, although the correct number of blank-character-separated fields will be written.

In list mode, standard output will not be buffered more than a line at a time.

If -v is specified in read, write or copy modes, pax will write the path names it processes to the standard error output using the following format:

```
%s
```

These path names will be written as soon as processing is begun on the file or archive member, and will be flushed to standard error. The trailing newline character, which will not be buffered, will be written when the file has been read or written.

If the -s option is specified, and the replacement string has a trailing p, substitutions will be written to standard error in the following format:

```
%s>>%s
```

In all operating modes of pax, optional messages of unspecified format concerning the input archive format and volume number, the number of files, blocks, volumes and media parts as well as other diagnostic messages may be written to standard error.

In all formats, for both standard output and standard error, it is unspecified how non-printable characters in path names or linknames are written.

If pax cannot create a file or a link when reading an archive or cannot find a file when writing an archive, or cannot preserve the user ID, group ID, or file mode when the -p option is specified, a diagnostic message will be written to standard error and a non-zero exit status will be returned, but processing will continue. In the case where pax cannot create a link to a file, pax will not, by default, create a second copy of the file.
If the extraction of a file from an archive is prematurely terminated by a signal or error, pax may have only partially extracted the file or (if the -n option was not specified) may have extracted a file of the same name as that specified by the user, but which is not the file the user wanted. Additionally, the file modes of extracted directories may have additional bits from the read, write, execute mask set as well as incorrect modification and access times.

**USAGE**

The -p (privileges) option was invented to reconcile differences between historical tar(1) and cpio(1) implementations. In particular, the two utilities use -m in diametrically opposed ways. The -p option also provides a consistent means of extending the ways in which future file attributes can be addressed, such as for enhanced security systems or high-performance files. Although it may seem complex, there are really two modes that will be most commonly used:

- **-p e**  
  “Preserve everything’’. This would be used by the historical superuser, someone with all the appropriate privileges, to preserve all aspects of the files as they are recorded in the archive. The e flag is the sum of o and p, and other implementation-dependent attributes.

- **-p p**  
  “Preserve’’ the file mode bits. This would be used by the user with regular privileges who wished to preserve aspects of the file other than the ownership. The file times are preserved by default, but two other flags are offered to disable these and use the time of extraction.

See largefile(5) for the description of the behavior of pax when encountering files greater than or equal to 2 Gbyte (2³¹ bytes).

**EXAMPLES**

**EXAMPLE 1** Copying the contents of the current directory

The following command:

```
example$ pax -w -f /dev/rmt/1m .
```

copies the contents of the current directory to tape drive 1, medium density. (This assumes historical System V device naming procedures. The historical BSD device name would be /dev/rmt9).

**EXAMPLE 2** Copying the directory hierarchy

The following commands:

```
example$ mkdir newdir
example$ pax -rw olddir newdir
```

copy the olddir directory hierarchy to newdir.
EXAMPLE 3 Reading an archive extracted relative to the current directory

The following command:

    example pax -r -s'user//*,,' -f a.pax

reads the archive a.pax, with all files rooted in /usr in the archive extracted relative to the current directory.

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of pax: LC_CTYPE, LC_MESSAGES, LC_TIME, and NLSPATH.

LC_COLLATE Determine the locale for the behaviour of ranges, equivalence classes, and multi-character collating elements used in the pattern matching expressions for the pattern operand, the basic regular expression for the -s option, and the extended regular expression defined for the yesexpr locale keyword in the LC_MESSAGES category.

EXIT STATUS

The following exit values are returned:

<table>
<thead>
<tr>
<th>Exit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>All files were processed successfully.</td>
</tr>
<tr>
<td>&gt;0</td>
<td>An error occurred.</td>
</tr>
</tbody>
</table>

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Stable</td>
</tr>
</tbody>
</table>

SEE ALSO

NAME perl – Practical Extraction and Report Language

SYNOPSIS perl [-sTuU] [-hv] [-V[:configvar]] [-cw] [-d[:debugger]]
        [-D[number/list]] [-pna] [-Ppattern] [-i[octal]] [-0[octal]] [-Idir]
        [-m[number/list]] [-M[' - ' module...'] [-P] [-S] [-x[dir]]
        [-i[extension]] [-e 'command'] [- -] [programfile [argument]...]

DESCRIPTION For ease of access, the Perl manual has been split up into several sections:

perl Perl overview (this section)
perlfaq Perl frequently asked questions
perltoc Perl documentation table of contents
perlbook Perl book information

perlsyn Perl syntax
perldata Perl data structures
perlop Perl operators and precedence
perlsub Perl subroutines
perlfunc Perl built-in functions
perlreftut Perl references short introduction
perldsc Perl data structures intro
perlrequick Perl regular expressions quick start
perlpod Perl plain old documentation
perlstyle Perl style guide
perltut Perl traps for the unwary

perlrun Perl execution and options
perldiag Perl diagnostic messages
perlexwarn Perl warnings and their control
perldebtut Perl debugging tutorial
perldebug Perl debugging

perlvar Perl predefined variables
perlpoi Perl data structures: arrays of arrays
perlogenutut Perl open() tutorial
perlreftut Perl regular expressions tutorial

perle Perl regular expressions, the rest of the story
perlref Perl references, the rest of the story

perlform Perl formats
perlboot Perl OO tutorial for beginners
perltoot Perl OO tutorial, part 1
perltootc Perl OO tutorial, part 2
perlobj Perl objects
perlbot Perl OO tricks and examples
perlnotie Perl objects hidden behind simple variables

perlipc Perl interprocess communication
perlfork Perl fork() information
perlnumber Perl number semantics
perlthrtut Perl threads tutorial

perlport Perl portability guide
perlllocale Perl locale support
perlunicode Perl unicode support
<table>
<thead>
<tr>
<th>perlsec</th>
<th>Perl security</th>
</tr>
</thead>
<tbody>
<tr>
<td>perlmod</td>
<td>Perl modules: how they work</td>
</tr>
<tr>
<td>perlmodlib</td>
<td>Perl modules: how to write and use</td>
</tr>
<tr>
<td>perlmodinstall</td>
<td>Perl modules: how to install from CPAN</td>
</tr>
<tr>
<td>perlnewmod</td>
<td>Perl modules: preparing a new module for distribution</td>
</tr>
<tr>
<td>perlfaq1</td>
<td>General Questions About Perl</td>
</tr>
<tr>
<td>perlfaq2</td>
<td>Obtaining and Learning about Perl</td>
</tr>
<tr>
<td>perlfaq3</td>
<td>Programming Tools</td>
</tr>
<tr>
<td>perlfaq4</td>
<td>Data Manipulation</td>
</tr>
<tr>
<td>perlfaq5</td>
<td>Files and Formats</td>
</tr>
<tr>
<td>perlfaq6</td>
<td>Regexes</td>
</tr>
<tr>
<td>perlfaq7</td>
<td>Perl Language Issues</td>
</tr>
<tr>
<td>perlfaq8</td>
<td>System Interaction</td>
</tr>
<tr>
<td>perlfaq9</td>
<td>Networking</td>
</tr>
<tr>
<td>perlcompile</td>
<td>Perl compiler suite intro</td>
</tr>
<tr>
<td>perlembed</td>
<td>Perl ways to embed perl in your C or C++ application</td>
</tr>
<tr>
<td>perldebguts</td>
<td>Perl debugging guts and tips</td>
</tr>
<tr>
<td>perlxstut</td>
<td>Perl XS tutorial</td>
</tr>
<tr>
<td>perlxs</td>
<td>Perl XS application programming interface</td>
</tr>
<tr>
<td>perlclib</td>
<td>Internal replacements for standard C library functions</td>
</tr>
<tr>
<td>perlinterp</td>
<td>Perl internal functions for those doing extensions</td>
</tr>
<tr>
<td>perlcall</td>
<td>Perl calling conventions from C</td>
</tr>
<tr>
<td>perlutil</td>
<td>utilities packaged with the Perl distribution</td>
</tr>
<tr>
<td>perlfilter</td>
<td>Perl source filters</td>
</tr>
<tr>
<td>perldbsfilter</td>
<td>Perl DBM filters</td>
</tr>
<tr>
<td>perlapitx</td>
<td>Perl API listing (autogenerated)</td>
</tr>
<tr>
<td>perlintern</td>
<td>Perl internal functions (autogenerated)</td>
</tr>
<tr>
<td>perlpio</td>
<td>Perl internal IO abstraction interface</td>
</tr>
<tr>
<td>perl_todo</td>
<td>Perl things to do</td>
</tr>
<tr>
<td>perlhack</td>
<td>Perl hackers guide</td>
</tr>
<tr>
<td>perlhist</td>
<td>Perl history records</td>
</tr>
<tr>
<td>perl5005delta</td>
<td>Perl changes since previous version</td>
</tr>
<tr>
<td>perl5004delta</td>
<td>Perl changes in version 5.005</td>
</tr>
<tr>
<td>perlsolaris</td>
<td>Perl notes for Solaris</td>
</tr>
</tbody>
</table>

(If you’re intending to read these straight through for the first time, the suggested order will tend to reduce the number of forward references.)

The manpages listed above are installed in the `/usr/perl5/man/` directory.

Extensive additional documentation for Perl modules is available. This additional documentation is also in the `/usr/local/lib/perl5/man` directory. Some of this additional documentation is distributed as standard with Perl, but you’ll also find documentation for any customer-installed third-party modules there.

Notice that running `catman(1M)` on the Perl manual pages is not supported. For other Solaris-specific details, see the NOTES section below.
You can also use the supplied /usr/perl5/bin/perldoc script to view Perl information.

If something strange has gone wrong with your program and you’re not sure where you should look for help, try the `-w` switch first. It will often point out exactly where the trouble is.

Perl is a language optimized for scanning arbitrary text files, extracting information from those text files, and printing reports based on that information. It’s also a good language for many system management tasks. The language is intended to be practical (easy to use, efficient, complete) rather than beautiful (tiny, elegant, minimal).

Perl combines (in the author’s opinion, anyway) some of the best features of C, `sed`, `awk`, and `sh`, so people familiar with those languages should have little difficulty with it. (Language historians will also note some vestiges of `csh`, Pascal, and even BASIC-PLUS.) Expression syntax corresponds quite closely to C expression syntax. Unlike most Unix utilities, Perl does not arbitrarily limit the size of your data—if you’ve got the memory, Perl can slurp in your whole file as a single string. Recursion is of unlimited depth. And the tables used by hashes (sometimes called “associative arrays”) grow as necessary to prevent degraded performance. Perl can use sophisticated pattern matching techniques to scan large amounts of data quickly. Although optimized for scanning text, Perl can also deal with binary data, and can make dbm files look like hashes. Setuid Perl scripts are safer than C programs through a dataflow tracing mechanism that prevents many stupid security holes.

If you have a problem that would ordinarily use `sed` or `awk` or `sh`, but it exceeds their capabilities or must run a little faster, and you don’t want to write the silly thing in C, then Perl may be for you. There are also translators to turn your `sed` and `awk` scripts into Perl scripts.

But wait, there’s more...

Began in 1993 (see the perlhist man page), Perl version 5 is nearly a complete rewrite that provides the following additional benefits:

- **Modularity and reusability using innumerable modules**
  Described in the perlmod man page, the perlmodlib man page, and the perlmodinstall man page.

- **Embeddable and Extensible**
  Described in the perlembed man page, the perlxstut man page, the perlxs man page, the percall man page, the perl guts man page, and the xsubpp man page.

- **Roll-your-own magic variables**
  (Including multiple simultaneous DBM implementations) Described in the perltie man page and the AnyDBM_File man page.

- **Subroutines can now be overridden,**
  autoloaded, and prototyped. Described in the perlsub man page.
• Arbitrarily nested data structures and anonymous functions. Described in the perlreftut man page, the perlref man page, the perldsc man page, and the perllol man page.

• Object-oriented programming
  Described in the perlobj man page, the perltoot man page, and the perlbot man page.

• Compilability into C code or Perl bytecode
  Described in the B man page and the B::Bytecode man page.

• Support for light-weight processes (threads)
  Described in the perlthrtut man page and the Thread man page. Notice that the Perl shipped as part of Solaris does NOT have threads support enabled. If you require threads support, you should build and install your own Perl version (see the NOTES section below).

• Support for internationalization, localization, and Unicode. Described in the perllocale man page and the utf8 man page.

• Lexical scoping
  Described in the perlsub man page.

• Regular expression enhancements
  Described in the perlrre man page, with additional examples in the perlop man page.

• Enhanced debugger and interactive Perl environment, with integrated editor support. Described in the perlddebug man page.

• POSIX 1003.1 compliant
  Described in the POSIX man page.

Okay, that’s definitely enough hype.

AVAILABILITY  Perl is available for most operating systems, including virtually all Unix-like platforms. See the Supported Platforms entry in the perlport man page for a listing.

ENVIRONMENT  The Perl shipped with Solaris is installed under /usr/perl5 rather than the default /usr/local location. This is so that it can coexist with a customer-installed Perl in the default /usr/local location.

Any additional modules that you choose to install will be placed in the /usr/perl5/site_perl/5.6.1 directory. The /usr/perl5/vendor_perl directory is reserved for SMI-provided modules.

Notice that the Perl utility scripts such as perldoc and perlbug are in the /usr/perl5/bin directory, so if you wish to use them you need to include /usr/perl5/bin in your PATH environment variable.

See also the perlrun man page.
perl(1)

AUTHOR  Larry Wall, with the help of oodles of other folks.

FILES  "@INC"  Locations of Perl libraries

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWp5u SUNWp5m</td>
</tr>
<tr>
<td></td>
<td>SUNWp5p</td>
</tr>
<tr>
<td></td>
<td>SUNWp5u SUNWp5m</td>
</tr>
<tr>
<td></td>
<td>SUNWp5p</td>
</tr>
<tr>
<td>Interface Stability</td>
<td></td>
</tr>
<tr>
<td>• Script interface</td>
<td>Evolving</td>
</tr>
<tr>
<td>• XSUB interface</td>
<td>Evolving</td>
</tr>
<tr>
<td>• Binary interface</td>
<td>Unstable</td>
</tr>
<tr>
<td>• Directory layout</td>
<td>Evolving</td>
</tr>
</tbody>
</table>

SEE ALSO  a2p  awk to perl translator

s2p  sed to perl translator

http://www.perl.com/  The Perl Home Page

http://www.perl.com/CPAN  The Comprehensive Perl Archive

DIAGNOSTICS  The "use warnings" pragma (and the -w switch) produces some lovely diagnostics.

See the perldiag man page for explanations of all Perl's diagnostics. The "use diagnostics" pragma automatically turns Perl's normally terse warnings and errors into these longer forms.

Compilation errors will tell you the line number of the error, with an indication of the next token or token type that was to be examined. (In a script passed to Perl via -e switches, each -e is counted as one line.)

Setuid scripts have additional constraints that can produce error messages such as "Insecure dependency". See the perlsec man page.

Did we mention that you should definitely consider using the -w switch?

NOTES  Solaris 9 contains two versions of Perl, 5.005_03 (as shipped in Solaris 8) and 5.6.1. /bin/perl is a link to the 5.6.1 interpreter, and /usr/perl5/bin is a link to the /usr/perl5/5.6.1/bin directory. It is likely that version 5.005_03 will be removed in a future release of Solaris.
Perl 5.6.1 has been built to be largefile-aware and to use 64-bit integers, although the interpreter itself is a 32-bit application (LP32). To view detailed configuration information, use `perl -V` and `perlbug -dv`.

Notice that 5.6.1 is binary incompatible with the 5.005_03 version, primarily due to the addition of largefile/64-bit integer support. Existing customer-installed XSUB-based modules will require recompilation, and non-XSUB modules will require reinstallation.

If you have any applications that require 5.005_03, you should make sure they explicitly use `/usr/perl5/5.005_03/bin/perl`. It is also possible to make 5.005_03 the default Perl version, although this is not recommended. The steps for this would be (as root):

```
# rm /usr/bin/perl
# ln -s ../perl5/5.00503/bin/perl /usr/bin/perl
# rm /usr/perl5/bin
# ln -s ./5.00503/bin /usr/perl5/bin
# rm /usr/perl5/man
# ln -s ./5.00503/man /usr/perl5/man
# rm /usr/perl5/pod
# ln -s ./5.00503/pod /usr/perl5/pod
```

If you wish to build and install your own version of Perl, you should NOT remove the 5.6.1 version of perl under `/usr/perl5`, as it is required by several system utilities. If you do not want to use the 5.005_03 version, you may remove that if you wish. The Perl package names are as follows:

<table>
<thead>
<tr>
<th>SUNWpl5u</th>
<th>Perl 5.6.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUNWpl5p</td>
<td>Perl 5.6.1 (POD Documentation)</td>
</tr>
<tr>
<td>SUNWpl5m</td>
<td>Perl 5.6.1 (Manual pages)</td>
</tr>
<tr>
<td>SUNWop5u</td>
<td>Perl 5.005_03</td>
</tr>
<tr>
<td>SUNWop5p</td>
<td>Perl 5.005_03 (POD Documentation)</td>
</tr>
<tr>
<td>SUNWop5m</td>
<td>Perl 5.005_03 (Manual pages)</td>
</tr>
</tbody>
</table>

The Perl motto is "There’s more than one way to do it." Divining how many more is left as an exercise to the reader.

The three principal virtues of a programmer are Laziness, Impatience, and Hubris. See the Camel Book for why.

**BUGS**

The `-w` switch is not mandatory.

Perl is at the mercy of your machine’s definitions of various operations such as type casting, ` atof ()`, and floating-point output with ` sprintf ()`.

If your stdio requires a seek or `eof` between reads and writes on a particular stream, so does Perl. (This doesn’t apply to `sysread()` and `syswrite()`.)
While none of the built-in data types have any arbitrary size limits (apart from memory size), there are still a few arbitrary limits: a given variable name may not be longer than 251 characters. Line numbers displayed by diagnostics are internally stored as short integers, so they are limited to a maximum of 65535 (higher numbers usually being affected by wraparound).

You may mail your bug reports (be sure to include full configuration information as output by the myconfig program in the Perl source tree, or by perl -v) to <perlbug@perl.org>.

Perl actually stands for Pathologically Eclectic Rubbish Lister, but don’t tell anyone I said that.
pfexec, pfsh, pfcsh, pfksh – execute a command in a profile

SYNOPSIS
/usr/bin/pfexec command
/usr/bin/pfsh [options] [argument ...]
/usr/bin/pfcsh [options] [argument ...]
/usr/bin/pfksh [options] [argument ...]

DESCRIPTION
The pfexec program is used to execute commands with the attributes specified by the user’s profiles in the exec_attr(4) database. It is invoked by the profile shells, pfsh, pfcsh, and pfksh which are linked to the Bourne shell, C shell, and Korn shell, respectively.

Profiles are searched in the order specified in the user’s entry in the user_attr(4) database. If the same command appears in more than one profile, the profile shell uses the first matching entry.

USAGE
pfexec is used to execute commands with predefined process attributes, such as specific user or group IDs.

Refer to the sh(1), csh(1), and ksh(1) man pages for complete usage descriptions of the profile shells.

EXIT STATUS
The following exit values are returned:
0 Successful completion.
1 An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO
csh(1), ksh(1), profiles(1), sh(1), exec_attr(4), prof_attr(4), user_attr(4), attributes(5)
The `pg` command is a filter that allows the examination of filenames one screenful at a time on a CRT. If the user types a RETURN, another page is displayed; other possibilities are listed below.

This command is different from previous paginators in that it allows you to back up and review something that has already passed. The method for doing this is explained below.

To determine terminal attributes, `pg` scans the `terminfo(4)` data base for the terminal type specified by the environment variable `TERM`. If `TERM` is not defined, the terminal type `dumb` is assumed.

### OPTIONS

- **-number**
  
  An integer specifying the size (in lines) of the window that `pg` is to use instead of the default. (On a terminal containing 24 lines, the default window size is 23).

- **-p string**
  
  `pg` uses string as the prompt. If the prompt string contains a `%d`, the first occurrence of `%d` in the prompt will be replaced by the current page number when the prompt is issued. The default prompt string is ”.”.

- **-c**
  
  Home the cursor and clear the screen before displaying each page. This option is ignored if `clear_screen` is not defined for this terminal type in the `terminfo(4)` data base.

- **-e**
  
  `pg` does not pause at the end of each file.

- **-f**
  
  Normally, `pg` splits lines longer than the screen width, but some sequences of characters in the text being displayed (for instance, escape sequences for underlining) generate undesirable results. The `-f` option inhibits `pg` from splitting lines.

- **-n**
  
  Normally, commands must be terminated by a `<newline>` character. This option causes an automatic end of command as soon as a command letter is entered.

- **-r**
  
  Restricted mode. The shell escape is disallowed. `pg` prints an error message but does not exit.

- **-s**
  
  `pg` prints all messages and prompts in the standard output mode (usually inverse video).

- **+linenumber**
  
  Start up at `linenumber`.

- **+/pattern/+**
  
  Start up at the first line containing the regular expression pattern.

### OPERANDS

The following operands are supported:

- `filename...`
filename

A path name of a text file to be displayed. If no filename is given, or if it is ~, the standard input is read.

The responses that may be typed when pg pauses can be divided into three categories: those causing further perusal, those that search, and those that modify the perusal environment.

Commands that cause further perusal normally take a preceding address, an optionally signed number indicating the point from which further text should be displayed. This address is interpreted in either pages or lines depending on the command. A signed address specifies a point relative to the current page or line, and an unsigned address specifies an address relative to the beginning of the file. Each command has a default address that is used if none is provided.

The perusal commands and their defaults are as follows:

(+1)<newline> or <blank> This causes one page to be displayed. The address is specified in pages.

(+1) l
With a relative address this causes pg to simulate scrolling the screen, forward or backward, the number of lines specified. With an absolute address this command prints a screenful beginning at the specified line.

(+1) d or ^D
Simulates scrolling half a screen forward or backward.

iE
Skip i screens of text.

iz
Same as <newline> except that i, if present, becomes the new default number of lines per screenful.

The following perusal commands take no address.

. or ^L
Typing a single period causes the current page of text to be redisplayed.

$
Displays the last windowful in the file. Use with caution when the input is a pipe.

The following commands are available for searching for text patterns in the text. The regular expressions are described on the regexp(5) manual page. They must always be terminated by a <newline>, even if the -n option is specified.

i/pattern/
Search forward for the i-th (default i=1) occurrence of pattern. Searching begins immediately after the current page and continues to the end of the current file, without wrap-around.

i*pattern*

i?pattern?
Search backwards for the i-th (default i=1) occurrence of pattern. Searching begins immediately before the current page and continues to the beginning of the current file, without
wrap-around. The ^ notation is useful for adds 100 terminals which will not properly handle the ?.

After searching, pg will normally display the line found at the top of the screen. This can be modified by appending n or b to the search command to leave the line found in the middle or at the bottom of the window from now on. The suffix t can be used to restore the original situation.

The user of pg can modify the environment of perusal with the following commands:

- `in` Begin perusing the ith next file in the command line. The i is an unsigned number, default value is 1.
- `ip` Begin perusing the ith previous file in the command line. i is an unsigned number, default is 1.
- `iw` Display another window of text. If i is present, set the window size to i.
- `s filename` Save the input in the named file. Only the current file being perused is saved. The white space between the s and filename is optional. This command must always be terminated by a <newline>, even if the -n option is specified.
- `h` Help by displaying an abbreviated summary of available commands.
- `q` or `Q` Quit pg.
- `!command` Command is passed to the shell, whose name is taken from the SHELL environment variable. If this is not available, the default shell is used. This command must always be terminated by a <newline>, even if the -n option is specified.

At any time when output is being sent to the terminal, the user can hit the quit key (normally CTRL-\) or the interrupt (break) key. This causes pg to stop sending output, and display the prompt. The user may then enter one of the above commands in the normal manner. Unfortunately, some output is lost when this is done, because any characters waiting in the terminal’s output queue are flushed when the quit signal occurs.

If the standard output is not a terminal, then pg acts just like cat(1), except that a header is printed before each file (if there is more than one).

**Large File Behavior**

See largefile(5) for the description of the behavior of pg when encountering files greater than or equal to 2 Gbyte (2^31 bytes).

**EXAMPLES**

**EXAMPLE 1** An example of the pg command.

The following command line uses pg to read the system news:

```
example% news | pg -p "(Page %d):"
```
ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of pg: LC_CTYPE, LC_MESSAGES, and NLSPATH.

The following environment variables affect the execution of pg:

- **COLUMNS**: Determine the horizontal screen size. If unset or NULL, use the value of TERM, the window size, baud rate, or some combination of these, to indicate the terminal type for the screen size calculation.

- **LINES**: Determine the number of lines to be displayed on the screen. If unset or NULL, use the value of TERM, the window size, baud rate, or some combination of these, to indicate the terminal type for the screen size calculation.

- **SHELL**: Determine the name of the command interpreter executed for a command.

- **TERM**: Determine terminal attributes. Optionally attempt to search a system-dependent database, keyed on the value of the TERM environment variable. If no information is available, a terminal incapable of cursor-addressable movement is assumed.

EXIT STATUS

The following exit values are returned:

- **0**: Successful completion.
- **>0**: An error occurred.

FILES

- `/tmp/pg*`: temporary file when input is from a pipe
- `/usr/share/lib/terminfo/?/*`: terminal information database

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

SEE ALSO

cat(1), grep(1), more(1), terminfo(4), attributes(5), environ(5), largefile(5), regex(5)

NOTES

While waiting for terminal input, pg responds to BREAK, CTRL-C, and CTRL-\ by terminating execution. Between prompts, however, these signals interrupt pg's current task and place the user in prompt mode. These should be used with caution when input is being read from a pipe, since an interrupt is likely to terminate the other commands in the pipeline.

The terminal /, ^, or ? may be omitted from the searching commands.
If terminal tabs are not set every eight positions, undesirable results may occur.

When using `pg` as a filter with another command that changes the terminal I/O options, terminal settings may not be restored correctly.
### NAME
pgrep, pkill – find or signal processes by name and other attributes

### SYNOPSIS

```
pgrep [-flvx] [-n | -o] [-d delim] [-P ppidlist] [-g pgrplist] [-s sidlist]
    [-T taskidlist] [pattern]
pkill [-signal] [-fvx] [-n | -o] [-P ppidlist] [-g pgrplist] [-s sidlist]
    [-T taskidlist] [pattern]
```

### DESCRIPTION
The **pgrep** utility examines the active processes on the system and reports the process IDs of the processes whose attributes match the criteria specified on the command line. Each process ID is printed as a decimal value and is separated from the next ID by a delimiter string, which defaults to a newline. For each attribute option, the user can specify a set of possible values separated by commas on the command line. For example,

```
pgrep -G other,daemon
```

matches processes whose real group ID is *other* OR *daemon*. If multiple criteria options are specified, **pgrep** matches processes whose attributes match the logical **AND** of the criteria options. For example,

```
pgrep -G other,daemon -U root,daemon
```

matches processes whose attributes are:

-(real group ID is other OR daemon) **AND**
-(real user ID is root OR daemon)

**pkill** functions identically to **pgrep**, except that each matching process is signaled as if by **kill**(1) instead of having its process ID printed. A signal name or number may be specified as the first command line option to **pkill**.

### OPTIONS
The following options are supported:

- **-d delim**
  Specifies the output delimiter string to be printed between each matching process ID. If no -d option is specified, the default is a newline character. The -d option is only valid when specified as an option to **pgrep**.

- **-f**
  The regular expression *pattern* should be matched against the full process argument string (obtained from the **pr_psargs** field of the **/proc/mmmn/psinfo** file). If no -f option is specified, the expression is matched only against the name of the executable file (obtained from the **pr_fname** field of the **/proc/mmmn/psinfo** file).

- **-g pgrplist**
  Matches only processes whose process group ID is in the given list. If group 0 is included in the list, this is interpreted as the process group ID of the **pgrep** or **pkill** process.
-G gidlist  Matches only processes whose real group ID is in the given list. Each group ID may be specified as either a group name or a numerical group ID.

-J projidlist Matches only processes whose project ID is in the given list. Each project ID may be specified as either a project name or a numerical project ID.

-l  Long output format. Prints the process name along with the process ID of each matching process. The process name is obtained from the pr_psargs or pr_fname field, depending on whether the -f option was specified (see above). The -l option is only valid when specified as an option to pgrep.

-n  Matches only the newest (most recently created) process that meets all other specified matching criteria. Cannot be used with option -o.

-o  Matches only the oldest (earliest created) process that meets all other specified matching criteria. Cannot be used with option -n.

-P ppidlist Matches only processes whose parent process ID is in the given list.

-s sidlist Matches only processes whose process session ID is in in the given list. If ID 0 is included in the list, this is interpreted as the session ID of the pgrep or pkill process.

-t termlist Matches only processes which are associated with a terminal in the given list. Each terminal is specified as the suffix following "/dev/" of the terminal's device path name in /dev. For example, term/a or pts/0.

-T taskidlist Matches only processes whose task ID is in the given list. If ID 0 is included in the list, this is interpreted as the task ID of the pgrep or pkill process.

-u euidlist Matches only processes whose effective user ID is in the given list. Each user ID may be specified as either a login name or a numerical user ID.

-U uidlist Matches only processes whose real user ID is in the given list. Each user ID may be specified as either a login name or a numerical user ID.

-v  Reverses the sense of the matching. Matches all processes except those which meet the specified matching criteria.

-x  Considers only processes whose argument string or executable file name exactly matches the specified pattern to be matching processes. The pattern match is considered to be exact when all characters in the process argument string or executable file name match the pattern.
-signal Specifies the signal to send to each matched process. If no signal is specified, SIGTERM is sent by default. The value of signal can be one of the symbolic names defined in signal(3HEAD) without the SIG prefix, or the corresponding signal number as a decimal value. The -signal option is only valid when specified as the first option to pkill.

**OPERANDS**

The following operand is supported:

*pattern* Specifies an Extended Regular Expression (ERE) pattern to match against either the executable file name or full process argument string. See regex(5) for a complete description of the ERE syntax.

**EXAMPLES**

**EXAMPLE 1** Obtaining a process ID

Obtain the process ID of sendmail:

```
example% psgrep -x -u root sendmail
```

**EXAMPLE 2** Terminating a process

Terminate the most recently created xterm:

```
example% pkill -n xterm
```

**EXIT STATUS**

The following exit values are returned:

0 One or more processes were matched.
1 No processes were matched.
2 Invalid command line options were specified.
3 A fatal error occurred.

**FILES**

/proc/nnnnn/psinfo Process information files

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

**SEE ALSO**

kill(1), proc(1), ps(1), truss(1), kill(2), signal(3HEAD), proc(4), attributes(5), regex(5)

**NOTES**

Both utilities match the ERE *pattern* argument against either the pr_fname or pr_psargs fields of the /proc/nnnnn/psinfo files. The lengths of these strings are limited according to definitions in <sys/procfs.h>. Patterns which can match strings longer than the current limits may fail to match the intended set of processes.
If the *pattern* argument contains ERE meta-characters which are also shell meta-characters, it may be necessary to enclose the pattern with appropriate shell quotes.

Defunct processes are never matched by either `pgrep` or `pkill`.

The current `pgrep` or `pkill` process will never consider itself a potential match.
NAME
pkginfo – display software package information

SYNOPSIS
pkginfo [-q | -x | -l] [-p | -i] [-r] [-a arch] [-v version]
       [-c category...] [pkginst...]
pkginfo [-d device] [-R root_path] [-q | -x | -l] [-a arch] [-v version]
       [-c category...] [pkginst...]

DESCRIPTION
pkginfo displays information about software packages that are installed on the
system (with the first synopsis) or that reside on a particular device or directory (with
the second synopsis).

Without options, pkginfo lists the primary category, package instance, and the names
of all completely installed and partially installed packages. It displays one line for
each package selected.

OPTIONS
The -p and -i options are meaningless if used in conjunction with the -d option.

The options -q, -x, and -l are mutually exclusive.

-a arch
   Specify the architecture of the package as arch.

-c category
   Display packages that match category. Categories are defined with the
   CATEGORY parameter in the pkginfo(4) file. If more than one
   category is supplied, the package needs to match only one
   category in the list. The match is not case specific.

-d device
   Defines a device, device, on which the software resides. device can
   be an absolute directory pathname or the identifiers for tape,
   floppy disk, removable disk, and so forth. The special token spool
   may be used to indicate the default installation spool directory
   (/var/spool/pkg).

-i
   Display information for fully installed packages only.

-l
   Specify long format, which includes all available information
   about the designated package(s).

-p
   Display information for partially installed packages only.

-q
   Do not list any information. Used from a program to check
   whether or not a package has been installed.

-r
   List the installation base for relocatable packages.

-R root_path
   Defines the full path name of a directory to use as the root_path. All
   files, including package system information files, are relocated to a
directory tree starting in the specified root_path.

-v version
   Specify the version of the package as version. The version is
   defined with the VERSION parameter in the pkginfo(4) file. All
   compatible versions can be requested by preceding the version
   name with a tilde (~). Multiple white spaces are replaced with a
   single white space during version comparison.
pkginfo(1)

-x Designate an extracted listing of package information. The listing contains the package abbreviation, package name, package architecture (if available) and package version (if available).

OPERANDS pkginst A package designation by its instance. An instance can be the package abbreviation or a specific instance (for example, inst.1 or inst.2). All instances of a package can be requested by inst.*. The asterisk character (*) is a special character to some shells and may need to be escaped. In the C-Shell, "*" must be surrounded by single quotes (’) or preceded by a backslash (\).

EXIT STATUS 0 Successful completion.
>0 An error occurred.

FILES /var/spool/pkg default installation spool directory

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO pkgtrans(1), pkgadd(1M), pkgask(1M), pkgchk(1M), pkgrm(1M), pkginfo(4), attributes(5)

Application Packaging Developer’s Guide
NAME
pkgmk – produce an installable package

SYNOPSIS
[-p pstamp] [-r root_path] [-v version] [variable=value...] [pkginstall]

DESCRIPTION
The pkgmk utility produces an installable package to be used as input to the
pkgadd(1M) command. The package contents will be in directory structure format.

The command uses the package prototype(4) file as input and creates a pkgmap(4)
file. The contents for each entry in the prototype file is copied to the appropriate
output location. Information concerning the contents (checksum, file size, modification
date) is computed and stored in the pkgmap file, along with attribute information
specified in the prototype file.

pkgmk searches for the files listed in the prototype(4) file as described in the
following conditions. Note: If a prototype file contains the explicit location of the file to
include in the package, then the following search explanations do not apply.

1. If neither -b nor -r options are specified, the file name component of each file path
   listed in the prototype(4) file is expected to be found in the same directory as the
   prototype(4) file

2. If -b is specified as a relative path (without a leading “/”), then base_src_dir is
   prepended to the relative file paths from the prototype(4) file. The resulting path
   is searched for in the root_path directories. If a root_path is not specified, it defaults
to “/”.

3. If -b is specified as an absolute path (with a leading “/”), then base_src_dir is
   prepended to the relative paths from the prototype(4) file and the result is the
   location of the file. root_path is not searched.

4. If -r is specified, then full file paths are used from the prototype(4) file. Relative
   paths have base_src_dir prepended. If base_src_dir is not specified, it defaults to “”.
   The resulting path is searched for in each directory of the root_path.

If you created your prototype file using “pkgproto a/relative/path” or
“pkgproto a/relative/path=install/path”, then you should use the -r
root_path option to specify the location of a/relative/path so that pkgmk can
correctly locate your source files.

OPTIONS
The following options are supported:

- a arch
  Overrides the architecture information provided in the
  pkginfo(4) file with arch.

- b base_src_dir
  Prepends the indicated base_src_dir to locate relocatable objects on
  the source machine. Use this option to search for all objects in the
  prototype file. pkgmk expects to find the objects in /base_src_dir or
  to locate the objects by use of the -b and -r options, respectively.

- d device
  Creates the package on device. device can be an absolute directory
  pathname or the identifiers for a floppy disk or removable disk
pkgmk(1)

(for example, /dev/diskette). The default device is the installation spool directory (/var/spool/pkg).

-`f prototype` Uses the file prototype as input to the command. The default prototype filename is [Pp]rototype.

-`l limit` Specifies the maximum size in 512 byte blocks of the output device as limit. By default, if the output file is a directory or a mountable device, pkgmk will employ the df(1M) command to dynamically calculate the amount of available space on the output device. This option is useful in conjunction with pkgtrans(1) to create a package with a datastream format.

-`ö` Overwrites the same instance; package instance will be overwritten if it already exists.

-`p pstamp` Overrides the production stamp definition in the pkginfo(4) file with pstamp.

-`r root_path` Uses the indicated root_path with the source pathname appended to locate objects on the source machine, using a comma (,) as the separator for the path elements. If this option is specified, look for the full destination path in each of the directories specified. If neither -b nor -r is specified, look for the leaf filename in the current directory.

-`v version` Overrides the version information provided in the pkginfo(4) file with version.

`variable=value` Places the indicated variable in the packaging environment. (See prototype(4) for definitions of variable specifications.)

OPERANDS
The following operand is supported:

`pkginst` A package designation by its instance. An instance can be the package abbreviation or a specific instance (for example, inst.1 or inst.2). All instances of a package can be requested by inst.*. The asterisk character (*) is a special character to some shells and may need to be escaped. In the C-Shell, "*" must be surrounded by single quotes (') or preceded by a backslash (\).

EXIT STATUS
The following exit values are returned:

0 Successful completion.

>0 An error occurred.

ATTRIBUTES
See attributes(5) for descriptions of the following attributes:
Architecture information is provided on the command line with the `-a` option or in the `prototype(4)` file. If no architecture information is supplied, `pkgmk` uses the output of `uname -m` (see `uname(1)`).

Version information is provided on the command line with the `-v` option or in the `pkginfo(4)` file. If no version information is supplied, a default based on the current date will be provided.

Command line definitions for both architecture and version override the `prototype(4)` definitions.

SEE ALSO

`pkgparam(1), pkgproto(1), pkgtrans(1), uname(1), df(1M), pkgadd(1M), pkginfo(4), pkgmap(4), prototype(4), attributes(5)`

NOTES

`Application Packaging Developer's Guide`
pkgparam(1)

NAME    pkgparam – display package parameter values

SYNOPSIS pkgparam [-v] [-d device] [-R root_path] pkginst [param...]

pkgparam -f filename [-v] [param...]

DESCRIPTION pkgparam displays the value associated with the parameter or parameters requested on the command line. The values are located in either the pkginfo(4) file for pkginst or from the specific file named with the -f option.

One parameter value is shown per line. Only the value of a parameter is given unless the -v option is used. With this option, the output of the command is in this format:

parameter1= 'value1'
parameter2= 'value2'
parameter3= 'value3'

If no parameters are specified on the command line, values for all parameters associated with the package are shown.

OPTIONS Options and arguments for this command are:

- d device Specify the device on which a pkginst is stored. It can be a directory pathname or the identifiers for tape, floppy disk, or removable disk (for example, /var/tmp, /dev/diskette, and /dev/dsk/c0d0s0). The special token spool may be used to represent the default installation spool directory (/var/spool/pkg).

- f filename Read filename for parameter values.

- R root_path Defines the full path name of a subdirectory to use as the root_path. All files, including package system information files, are relocated to a directory tree starting in the specified root_path.

- v Verbose mode. Display name of parameter and its value.

OPERANDS pkginst Defines a specific package instance for which parameter values should be displayed.

param Defines a specific parameter whose value should be displayed.

ERRORS If parameter information is not available for the indicated package, the command exits with a non-zero status.

EXIT STATUS 0 Successful completion.

>0 An error occurred.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:
SEE ALSO
pkgmk(1), pkgproto(1), pkgtrans(1), pkginfo(4), attributes(5)

Application Packaging Developer’s Guide

NOTES
With the -f option, you can specify the file from which parameter values should be extracted. This file should be in the same format as a pkginfo(4) file. For example, such a file might be created during package development and used while testing software during this stage.
pkgproto(1)

NAME    pkgproto – generate prototype file entries for input to pkgmk command

SYNOPSIS pkgproto [-i] [-c class] [path1]
 pkgproto [-i] [-c class] [path1=path2...]

DESCRIPTION pkgproto scans the indicated paths and generates prototype(4) file entries that
may be used as input to the pkgmk(1) command.

If no paths are specified on the command line, standard input is assumed to be a list of
paths. If the pathname listed on the command line is a directory, the contents of the
directory is searched. However, if input is read from stdin, a directory specified as a
pathname will not be searched.

OPTIONS -i           Ignores symbolic links and records the paths as ftype=f (a file)
                      versus ftype=s (symbolic link).
- c class            Maps the class of all paths to class.

OPERANDS path1      Pathname where objects are located.
 path2      Pathname which should be substituted on output for path1.

EXAMPLES EXAMPLE 1 Examples of the use of pkgproto.1.

The following two examples show uses of pkgproto and a partial listing of the
output produced.

Example 1:

example% pkgproto /bin=bin /usr/bin=usrbin /etc=etc
f none bin/sed=/bin/sed 0775 bin bin
f none bin/sh=/bin/sh 0755 bin daemon
f none bin/sort=/bin/sort 0755 bin bin
f none usrb/bin/sdb=/usr/bin/sdb 0775 bin bin
f none usrb/bin/sh1=/usr/bin/sh1 4755 bin bin
d none etc/master.d 0755 root daemon
f none etc/master.d/kernel=/etc/master.d/kernel 0644 root daemon
cp none etc/rc=/etc/rc 0744 root daemon

Example 2:

example% find / -type d -print | pkgproto
D none / 755 root root
D none /bin 755 bin bin
D none /usr 755 root root
D none /usr/bin 775 bin bin
D none /etc 755 root root
D none /tmp 777 root root

EXIT STATUS  0      Successful completion.
             >0      An error occurred.

ATTRIBUTES    See attributes(5) for descriptions of the following attributes:
By default, pkgproto creates symbolic link entries for any symbolic link encountered (ftype=s). When you use the -i option, pkgproto creates a file entry for symbolic links (ftype=f). The prototype(4) file would have to be edited to assign such file types as v (volatile), e (editable), or x (exclusive directory). pkgproto detects linked files. If multiple files are linked together, the first path encountered is considered the source of the link.

By default, pkgproto prints prototype entries on the standard output. However, the output should be saved in a file (named Prototype or prototype, for convenience) to be used as input to the pkgmk(1) command.

SEE ALSO  pkgmk(1), pkgparam(1), pkgtrans(1), prototype(4), attributes(5)

NOTES  Application Packaging Developer’s Guide

By default, pkgproto creates symbolic link entries for any symbolic link encountered (ftype=s). When you use the -i option, pkgproto creates a file entry for symbolic links (ftype=f). The prototype(4) file would have to be edited to assign such file types as v (volatile), e (editable), or x (exclusive directory). pkgproto detects linked files. If multiple files are linked together, the first path encountered is considered the source of the link.

By default, pkgproto prints prototype entries on the standard output. However, the output should be saved in a file (named Prototype or prototype, for convenience) to be used as input to the pkgmk(1) command.
pkgtrans(1)

NAME  pkgtrans – translate package format

SYNOPSIS  pkgtrans [-inos] device1 device2 [pkginst...]

DESCRIPTION  pkgtrans translates an installable package from one format to another. It translates:
  ■ a file system format to a datastream
  ■ a datastream to a file system format
  ■ one file system format to another file system format

OPTIONS  The options and arguments for this command are:
  -i  Copy only the pkginfo(4) and pkgmap(4) files.
  -n  Create a new instance of the package on the destination device if any instance of this package already exists, up to the number specified by the MAXINST variable in the pkginfo(4) file.
  -o  Overwrite the same instance on the destination device; package instance will be overwritten if it already exists.
  -s  Indicates that the package should be written to device2 as a datastream rather than as a file system. The default behavior is to write a file system format on devices that support both formats.

OPERANDS  
  device1  Indicates the source device. The package or packages on this device will be translated and placed on device2.
  device2  Indicates the destination device. Translated packages will be placed on this device.
  pkginst  Specifies which package instance or instances on device1 should be translated. The token all may be used to indicate all packages. pkginst.* can be used to indicate all instances of a package. If no packages are defined, a prompt shows all packages on the device and asks which to translate.

  The asterisk character (*) is a special character to some shells and may need to be escaped. In the C-Shell, "*" must be surrounded by single quotes (’) or preceded by a backslash (\).

EXAMPLES  EXAMPLE 1 Examples of the pkgtrans command.

The following example translates all packages on the floppy drive /dev/diskette and places the translations on /tmp:

  example$ pkgtrans /dev/diskette /tmp all

The next example translates packages pkg1 and pkg2 on /tmp and places their translations (that is, a datastream) on the 9track1 output device:

  example$ pkgtrans /tmp 9track1 pkg1 pkg2
EXAMPLE 1 Examples of the pkgtrans command. (Continued)

The next example translates pkg1 and pkg2 on /tmp and places them on the diskette in a datastream format:

eexample% pkgtrans -s /tmp /dev/diskette pkg1 pkg2

ENVIRONMENT VARIABLES

The MAXINST variable is set in the pkginfo(4) file and declares the maximum number of package instances.

EXIT STATUS

0 Successful completion.

>0 An error occurred.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO

pkginfo(1), pkgmk(1), pkgparam(1), pkgproto(1), installf(1M), pkgadd(1M), pkgask(1M), pkgrm(1M), removef(1M), pkginfo(4), pkgmap(4), attributes(5)

Application Packaging Developer's Guide

NOTES

Device specifications can be either the special node name (for example, /dev/diskette) or a device alias (for example, diskette1). The device spool indicates the default spool directory. Source and destination devices cannot be the same.

By default, pkgtrans will not translate any instance of a package if any instance of that package already exists on the destination device. Using the -n option creates a new instance if an instance of this package already exists. Using the -o option overwrites an instance of this package if it already exists. Neither of these options are useful if the destination device is a datastream.
NAME
plimit – get or set the resource limits of running processes

SYNOPSIS
plimit [-km] pid...
plimit {-cdfnstv} soft,hard... pid...

DESCRIPTION
If one or more of the cdfnstv options is specified, plimit sets the soft (current) limit and/or the hard (maximum) limit of the indicated resource(s) in the processes identified by the process-ID list, pid. Otherwise plimit reports the resource limits of the processes identified by the process-ID list, pid.

Only the owner of a process or the super-user is permitted either to get or to set the resource limits of a process. Only the super-user can increase the hard limit.

OPTIONS
The following options are supported:
-\k On output, show file sizes in kilobytes (1024 bytes) rather than in 512-byte blocks.
-\m On output, show file and memory sizes in megabytes (1024*1024 bytes).

The remainder of the options are used to change specified resource limits. They each accept an argument of the form:

soft,hard

where soft specifies the soft (current) limit and hard specifies the hard (maximum) limit. If the hard limit is not specified, the comma may be omitted. If the soft limit is an empty string, only the hard limit is set. Each limit is either the literal string unlimited, or a number, with an optional scaling factor, as follows:

nk n kilobytes
mm n megabytes (minutes for CPU time)
nh n hours (for CPU time only)
mm:ss minutes and seconds (for CPU time only)

The soft limit cannot exceed the hard limit.

-\c soft,hard Set core file size limits (default unit is 512-byte blocks).
-\d soft,hard Set data segment (heap) size limits (default unit is kilobytes).
-\f soft,hard Set file size limits (default unit is 512-byte blocks).
-\n soft,hard Set file descriptor limits (no default unit).
-\s soft,hard Set stack segment size limits (default unit is kilobytes).
-\t soft,hard Set CPU time limits (default unit is seconds).
-\v soft,hard Set virtual memory size limits (default unit is kilobytes).
plimit(1)

OPERANDS

The following operands are supported.

pid Process ID list.

EXIT STATUS

plimit returns the exit value zero on success, non-zero on failure (such as no such
process, permission denied, or invalid option).

FILES

/proc/pid/* process information and control files

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
</tbody>
</table>

SEE ALSO

ulimit(1), proc(1), getrlimit(2), setrlimit(2), proc(4), attributes(5),
The `plot` utility reads plotting instructions (see `plot(4B)`) from the standard input and produces plotting instructions suitable for a particular terminal on the standard output.

If no terminal is specified, the environment variable `TERM` is used. The default terminal is `tek`.

Except for `ver`, the following terminal-types can be used with `lpr-g` (see `lpr(1B)`) to produce plotted output:

- 2648
- `2648a` | 2648 | h8 | hp2648 | hp2648a
  Hewlett Packard 2648 graphics terminal.
- `hp7221` | hp7 | h7 |
  Hewlett Packard 7221 plotter.
- 300
  DASI 300 or GSI terminal (Diablo mechanism).
- `300s` | `300S`
  DASI 300s terminal (Diablo mechanism).
- 450
  DASI Hyterm 450 terminal (Diablo mechanism).
- `4013`
  Tektronix 4013 storage scope.
- `4014` | `tek`
  Tektronix 4014 and 4015 storage scope with Enhanced Graphics Module. (Use 4013 for Tektronix 4014 or 4015 without the Enhanced Graphics Module).
- `aed`
  AED 512 color graphics terminal.
- `bgplot` | `bitgraph`
  BBN bitgraph graphics terminal.
- `crt`
  Any crt terminal capable of running vi(1).
- `dumb` | `un` | `unknown`
  Dumb terminals without cursor addressing or line printers.
- `gigi` | `vt125`
  DEC vt125 terminal.
- `implot`
  Imagen plotter.
var
  Benson Varian printer-plotter
ver
  Versatec D1200A printer-plotter. The output is scan-converted and suitable input to
  'lpr -v'.

FILES
/usr/ucb/aedplot
/usr/ucb/atoplot
/usr/ucb/bgplot
/usr/ucb/crtplot
/usr/ucb/dumbplot
/usr/ucb/gigiplot
/usr/ucb/hp7221plot
/usr/ucb/hpplot
/usr/ucb/implot
/usr/ucb/plot
/usr/ucb/plottoa
/usr/ucb/t300
/usr/ucb/t300s
/usr/ucb/t4013
/usr/ucb/t450
/usr/ucb/tek
/usr/ucb/vplot

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

SEE ALSO  graph(1), tplot(1), vi(1), lpr(1B), plot(4B), attributes(5)
pmap(1)

NAME
pmap – display information about the address space of a process

SYNOPSIS
/usr/bin/pmap [-rslF] [pid | core...]
/usr/bin/pmap -x [-s1F] [pid | core...]
/usr/bin/pmap -S [-1F] [pid | core...]

DESCRIPTION
The pmap utility prints information about the address space of a process.

OPTIONS
The following options are supported:

-a  Prints anonymous and swap reservations for shared mappings.
-F  Force. Grabs the target process even if another process has control.
-r  Prints the process’s reserved addresses.
-s  Prints HAT page size information.

USAGE
The pmap utility prints information about the address space of a process.

Process Mappings
/usr/bin/pmap [-rslF] [pid | core ] ...

By default, pmap displays the mappings in the virtual address order they are
mapped into the process. The mapping size, flags and mapped object name are
shown.

Process anon/locked mapping details
/usr/bin/pmap -x [-s1F] [pid | core ] ...

The -x option displays additional information per mapping. The size of each
mapping, the amount of resident physical memory, the amount of anonymous
memory, and the amount of memory locked is shown with this option.

Swap Reservations
/usr/bin/pmap -S [-1F] [pid | core ] ...

The -S option displays swap reservation information per mapping.

DISPLAY FORMATS
One line of output is printed for each mapping within the process, unless the -s
option is specified, where one line is printed for a contiguous mapping of each
hardware translation page size.

Virtual Address
The first column of output represents the starting virtual address of each mapping.
Virtual addresses are displayed in ascending order.

Virtual Mapping Size
The virtual size of each mapping.
Resident Physical Memory
The amount of physical memory resident for each mapping, including that which is
shared with other address spaces.

Anonymous Memory
The amount of anonymous memory is reported for each mapping. Anonymous
memory shared with other address spaces is not included, unless the -a option is
specified.

Anonymous memory is reported for the process heap, stack, for ‘copy on write’
pages with mappings mapped with MAP_PRIVATE.

Locked
The number of pages locked within the mapping. Typical examples are memory
locked with mlock() and System V shared memory created with
SHM_SHARE_MMU.

Permissions/Flags
The virtual memory permissions are shown for each mapping. Valid permissions
are:

r: The mapping may be read by the process.
w: The mapping may be written by the process.
x: Instructions that reside within the mapping may be executed by the
process.

Flags showing additional information for each mapping may be displayed:
s: The mapping is shared such that changes made in the observed address
space are committed to the mapped file, and are visible from all other
processes sharing the mapping.

R: Swap space is not reserved for this mapping. Mappings created with
MAP_NORESERVE and System V ISM shared memory mappings do not
reserve swap space.

Mapping Name
A descriptive name for each mapping. The following major types of names are
displayed for mappings:

- A mapped file: For mappings between a process and a file, the pmap command
  attempts to resolve the file name for each mapping. If the file name cannot be
  resolved, pmap displays the major and minor number of the device containing
  the file, and the file system inode number of the file.

- Anonymous memory: Memory not relating to any named object or file within the
  file system is reported as [ anon ].

The pmap command displays common names for certain known anonymous
memory mappings, such as:

[ heap ] The process heap.
The process stack.

If the common name for the mapping is unknown, pmap displays [ anon ] as the mapping name.

- **System V Shared Memory:** Mappings created using System V shared memory system calls are reported with the names shown below:
  - `shmid=n:` The mapping is a System V shared memory mapping. The shared memory identifier that the mapping was created with is reported.
  - `ism shmid=n:` The mapping is an "Intimate Shared Memory" variant of System V shared memory. ISM mappings are created with the `SHM_SHARE_MMU` flag set, in accordance with `shmat(2)` (see `shmop(2)`).
  - `dism shmid=n:` The mapping is a pageable variant of ISM. Pageable ISM is created with the `SHM_PAGEABLE` flag set in accordance with `shmat(2)` (see `shmop(2)`).

- **Other:** Mappings of other objects, including devices such as frame buffers. No mapping name is shown for other mapped objects.

### EXAMPLES

**EXAMPLE 1** Displaying process mappings

By default, pmap prints one line for each mapping within the address space of the target process. The following example displays the address space of a typical bourne shell:

```bash
example$ pmap 102905
102905:  sh
  00010000 192K r-x-- /usr/bin/ksh
  00040000  8K rwx-- /usr/bin/ksh
  00042000  40K rwx-- [ heap ]
  FF180000  664K r-x-- /usr/lib/libc.so.1
  FF236000  24K rwx-- /usr/lib/libc.so.1
  FF23C000  8K rwx-- /usr/lib/libc.so.1
  FF250000  8K rwx-- [ anon ]
  FF260000 16K r-x-- /usr/lib/en_US.ISO8859-1.so.2
  FF272000 16K rwx-- /usr/lib/en_US.ISO8859-1.so.2
  FF280000 560K r-x-- /usr/lib/libnsl.so.1
  FF31C000  32K rwx-- /usr/lib/libnsl.so.1
  FF324000  32K rwx-- /usr/lib/libnsl.so.1
  FF340000 16K r-x-- /usr/lib/libc_psr.so.1
  FF350000 16K r-x-- /usr/lib/libmp.so.2
  FF364000  8K rwx-- /usr/lib/libmp.so.2
  FF380000  40K r-x-- /usr/lib/libsocket.so.1
  FF39A000  8K rwx-- /usr/lib/libsocket.so.1
  FF3A0000  8K r-x-- /usr/lib/libdl.so.1
  FF3B0000  8K rwx-- [ anon ]
  FF3C0000 152K r-x-- /usr/lib/ld.so.1
  FF3F6000  8K rwx-- /usr/lib/ld.so.1
  FFBFC000  16K r-x-- [ stack ]
total 1880K
```
EXAMPLE 2 Displaying memory allocation and mapping types

The -x option can be used to provide information about the memory allocation and mapping types per mapping. The amount of resident, non-shared anonymous, and locked memory is shown for each mapping:

```
example$ pmap -x 102908
102908: sh
Address       Kbytes Resident Anon Locked Mode Mapped File
00010000     88  88   -    -    r-x--    sh
00036000      8   8    8    -    rwx--    sh
00038000    16  16   16    -    -    rwx-- [ heap ]
FF260000    16  16    -    -    r-x--  en_US.ISO8859-1.so.2
FF272000    16  16    -    -    rwx--  en_US.ISO8859-1.so.2
FF280000   664 624    -    -    r-x--  libc.so.1
FF366000    32  32    8    -    rwx--  libc.so.1
FF368000   16  16    -    -    r-x-- libc_psr.so.1
FF380000    24  24    -    -    r-x--  libgen.so.1
FF396000    8   8    -    -    rwx--  libgen.so.1
FF3A0000    8   8    -    -    r-x--  libdl.so.1
FF3B0000    8   8    -    -    r-x--  libdl.so.1
FF3C0000 152 152    -    -    r-x-- [ anon ]
FF3FC000 152 152    -    -    r-x--  ld.so.1
FF3FD000    8   8    8    -    rwx--  ld.so.1
FF3EF000    8   8    8    -    rwx-- [ stack ]
----------------- ----- ----- ----- -----
total Kb 1072 1032 56 -
```

The amount of incremental memory used by each additional instance of a process can be estimated by using the resident and anonymous memory counts of each mapping.

In the above example, the bourne shell has a resident memory size of 1032Kbytes. However, a large amount of the physical memory used by the shell is shared with other instances of shell. Another identical instance of the shell will share physical memory with the other shell where possible, and allocate anonymous memory for any non-shared portion. In the above example, each additional bourne shell uses approximately 56Kbytes of additional physical memory.

A more complex example shows the output format for a process containing different mapping types. In this example, the mappings are as follows:

```
00010000: Executable text, mapped from 'maps' program
00020000: Executable data, mapped from 'maps' program
00022000: Program heap
00030000: A mapped file, mapped MAP_SHARED
00040000: A mapped file, mapped MAP_PRIVATE
00500000: A mapped file, mapped MAP_PRIVATE | MAP_NORESERVE
00600000: Anonymous memory, created by mapping /dev/zero
00700000: Anonymous memory, created by mapping /dev/zero
```
**EXAMPLE 2** Displaying memory allocation and mapping types

(Continued)

with MAP_NORESERVE

0800000: A DISM shared memory mapping, created with SHM_PAGEABLE
with 8MB locked via mlock(2)

0900000: A DISM shared memory mapping, created with SHM_PAGEABLE,
with 4MB of its pages touched.

0A00000: A DISM shared memory mapping, created with SHM_PAGEABLE,
with none of its pages touched.

0B00000: An ISM shared memory mapping, created with SHM_SHARE_MMU

```bash
e.example$ pmap -xs 15492
```

```plaintext
Address Kbytes  RSS  Anon  Locked  Mode   Mapped   File
00010000   8   8    -  -    r-x--   maps
00020000   8   8    8  -    rwxs--   maps
00022000   20344 16248 16248  -  rwxs--  [ heap ]
03000000  1024 1024   -  -    rw-s    dev:0,2  ino:4628487
04000000  1024 1024  512  -    rw---   dev:0,2  ino:4628487
05000000  1024 1024  512  -    rw--R   dev:0,2  ino:4628487
06000000  1024 1024  1024  -  rw----  [ anon ]
07000000  512  512   512  -    rw--R   [ anon ]
08000000  8192  8192  -  8192  rwxs-  [ dism shmid=0x5]
09000000  8192  4096  -  -    rwxs-  [ dism shmid=0x4]
0A000000  8192  8192  -  8192  rwxsR  [ ism shmid=0x2]
0B000000  8192  8192  -  8192  rwxsR  [ ism shmid=0x3]
FF280000  680  672  -  -    r-x--   libc.so.1
FF33A000  32   32   32  -    rwxs--  libc.so.1
FF390000  8   8  -  -    r-x--   libc_psr.so.1
FF3A0000  8  8   8  -    rwxs--  libdl.so.1
FF3B0000  8   8   8  -    rwxs--  [ anon ]
FF3C0000  122  122  -  -    r-x--   ld.so.1
FF3F6000  152  152  -  -    r-x--   ld.so.1
FF4F0000  24   24   24  -    rwxs--  [ stack ]
```

**EXAMPLE 3** Displaying Page Size Information

The `-s` option can be used to display the hardware translation page sizes for each portion of the address space. (See `memcntl(2)` for further information on Solaris multiple page size support).

In the example below, we can see that the majority of the mappings are using an 8K-Byte page size, while the heap is using a 4M-Byte page size.

Notice that non-contiguous regions of resident pages of the same page size are reported as separate mappings. In the example below, the `libc.so` library is reported as separate mappings, since only some of the `libc.so` text is resident.
### EXAMPLE 3 Displaying Page Size Information

**Continued**

```
example$ pmap -xs 15492

15492: ./maps
```

<table>
<thead>
<tr>
<th>Address</th>
<th>Kbytes</th>
<th>RSS</th>
<th>Anon</th>
<th>Locked</th>
<th>Pgsz</th>
<th>Mode</th>
<th>Mapped</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>00010000</td>
<td>8</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>8K</td>
<td>r-x--</td>
<td></td>
</tr>
<tr>
<td>00020000</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>8K</td>
<td>rwx--</td>
<td>maps</td>
</tr>
<tr>
<td>00022000</td>
<td>3960</td>
<td>3960</td>
<td>3960</td>
<td>-</td>
<td>8K</td>
<td>rwx--</td>
<td>[ heap ]</td>
<td></td>
</tr>
<tr>
<td>00400000</td>
<td>8192</td>
<td>8192</td>
<td>8192</td>
<td>-</td>
<td>4M</td>
<td>rwx--</td>
<td>[ heap ]</td>
<td></td>
</tr>
<tr>
<td>00C00000</td>
<td>4096</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>rwx--</td>
<td>[ heap ]</td>
<td></td>
</tr>
<tr>
<td>01000000</td>
<td>4096</td>
<td>4096</td>
<td>4096</td>
<td>-</td>
<td>-</td>
<td>4M</td>
<td>rwx--</td>
<td></td>
</tr>
<tr>
<td>03000000</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
<td>8K</td>
<td>r-w-s</td>
<td>dev:0,2</td>
<td>ino:4628487</td>
<td></td>
</tr>
<tr>
<td>04000000</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>8K</td>
<td>r-w-s</td>
<td>dev:0,2</td>
<td>ino:4628487</td>
<td></td>
</tr>
<tr>
<td>04080000</td>
<td>512</td>
<td>512</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>r-w-s</td>
<td>dev:0,2</td>
<td>ino:4628487</td>
</tr>
<tr>
<td>05000000</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>8K</td>
<td>r-w-s</td>
<td>dev:0,2</td>
<td>ino:4628487</td>
<td></td>
</tr>
<tr>
<td>05080000</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>8K</td>
<td>r-w-s</td>
<td>dev:0,2</td>
<td>ino:4628487</td>
<td></td>
</tr>
<tr>
<td>06000000</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
<td>8K</td>
<td>r-w-s</td>
<td>[ anon ]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>07000000</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>8K</td>
<td>r-w-s</td>
<td>[ anon ]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>08000000</td>
<td>8192</td>
<td>8192</td>
<td>8192</td>
<td>-</td>
<td>8K</td>
<td>r-wxs</td>
<td>[ dism shmid=0x5 ]</td>
<td></td>
</tr>
<tr>
<td>09000000</td>
<td>4096</td>
<td>4096</td>
<td>4096</td>
<td>-</td>
<td>8K</td>
<td>r-wxs</td>
<td>[ dism shmid=0x4 ]</td>
<td></td>
</tr>
<tr>
<td>0A000000</td>
<td>4096</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>r-wxs</td>
<td>[ dism shmid=0x2 ]</td>
<td></td>
</tr>
<tr>
<td>0B000000</td>
<td>8192</td>
<td>8192</td>
<td>8192</td>
<td>4M</td>
<td>r-wxsR</td>
<td>[ ism shmid=0x3 ]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FF280000</td>
<td>136</td>
<td>136</td>
<td>-</td>
<td>8K</td>
<td>r-x--</td>
<td>libc.so.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FF2A2000</td>
<td>120</td>
<td>120</td>
<td>-</td>
<td>-</td>
<td>r-x--</td>
<td>libc.so.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FF2C0000</td>
<td>128</td>
<td>128</td>
<td>-</td>
<td>8K</td>
<td>r-x--</td>
<td>libc.so.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FF2E0000</td>
<td>200</td>
<td>200</td>
<td>-</td>
<td>-</td>
<td>r-x--</td>
<td>libc.so.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FF312000</td>
<td>48</td>
<td>48</td>
<td>-</td>
<td>8K</td>
<td>r-x--</td>
<td>libc.so.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FF3A0000</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>8K</td>
<td>r-wxs</td>
<td>libc.so.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FF390000</td>
<td>8</td>
<td>8</td>
<td>-</td>
<td>8K</td>
<td>r-x--</td>
<td>libc_psr.so.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FF3A0000</td>
<td>8</td>
<td>8</td>
<td>-</td>
<td>8K</td>
<td>r-x--</td>
<td>libdl.so.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FF3B0000</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8K</td>
<td>r-wxs</td>
<td>[ anon ]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FF3C0000</td>
<td>152</td>
<td>152</td>
<td>-</td>
<td>8K</td>
<td>r-x--</td>
<td>ld.so.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FF3F6000</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8K</td>
<td>r-wxs</td>
<td>ld.so.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFF8A000</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>-</td>
<td>8K</td>
<td>r-wxs</td>
<td>[ stack ]</td>
<td></td>
</tr>
</tbody>
</table>

```

### EXAMPLE 4 Displaying swap reservations

The `-S` option can be used to display the swap reservations for a process. The amount of swap space reserved is displayed for each mapping within the process. Swap reservations are reported as zero for shared mappings, since they are accounted for only once system wide.

```
example$ pmap -S 15492

15492: ./maps
```

<table>
<thead>
<tr>
<th>Address</th>
<th>Kbytes</th>
<th>Swap</th>
<th>Mode</th>
<th>Mapped</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>00010000</td>
<td>8</td>
<td>-</td>
<td>r-x--</td>
<td>maps</td>
<td></td>
</tr>
<tr>
<td>00020000</td>
<td>8</td>
<td>8</td>
<td>rwx--</td>
<td>maps</td>
<td></td>
</tr>
<tr>
<td>00022000</td>
<td>20344</td>
<td>20344</td>
<td>rwx--</td>
<td>[ heap ]</td>
<td></td>
</tr>
<tr>
<td>03000000</td>
<td>1024</td>
<td>1024</td>
<td>r-w-s</td>
<td>dev:0,2</td>
<td>ino:4628487</td>
</tr>
<tr>
<td>04000000</td>
<td>1024</td>
<td>1024</td>
<td>r-w-s</td>
<td>dev:0,2</td>
<td>ino:4628487</td>
</tr>
<tr>
<td>05000000</td>
<td>1024</td>
<td>512</td>
<td>r-w-s</td>
<td>dev:0,2</td>
<td>ino:4628487</td>
</tr>
<tr>
<td>06000000</td>
<td>1024</td>
<td>1024</td>
<td>r-w-s</td>
<td>[ anon ]</td>
<td></td>
</tr>
</tbody>
</table>
EXAMPLE 4 Displaying swap reservations  (Continued)

```
07000000  512  512  rw--R  [ anon ]
08000000  8192 -  rwxs-  [ dism shmid=0x5]
09000000  8192 -  rwxs-  [ dism shmid=0x4]
0A000000  8192 -  rwxs-  [ dism shmid=0x2]
0B000000  8192 -  rwxsR [ ism shmid=0x3]
FF280000  680 -  r-x--  libc.so.1
FF33A000  32  32  rwx--  libc.so.1
FF390000  8 -  r-x--  libc_psr.so.1
FF3A0000  8 -  r-x--  libdl.so.1
FF3B0000  8  8  rwx--  [ anon ]
FF3C0000  152 -  r-x--  ld.so.1
FF3F6000  8  8  rwx--  [ stack ]
-------- ------- -------
total Kb  50464  23496
```

The swap reservation information can be used to estimate the amount of virtual swap used by each additional process. Each process consumes virtual swap from a global virtual swap pool. Global swap reservations are reported by the ‘avail’ field of the swap(1M) command.

EXIT STATUS
The following exit values are returned:

-  0  Successful operation.
- non-zero  An error has occurred.

FILES
/proc/*  process files
/usr/proc/lib/*  proc tools supporting files

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

<table>
<thead>
<tr>
<th>Attribute Type</th>
<th>Attribute Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu (32-bit)</td>
</tr>
<tr>
<td></td>
<td>SUNWesxu (64-bit)</td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Evolving</td>
</tr>
<tr>
<td>Command Syntax</td>
<td>Unstable</td>
</tr>
</tbody>
</table>

SEE ALSO
ldd(1), mdb(1), proc(1), ps(1), swap(1M), memcntl(2), shmop(2), dlopen(3DL), proc(4), attributes(5)
### NAME
postdaisy – PostScript translator for Diablo 630 daisy-wheel files

### SYNOPSIS
```
postdaisy [-c num] [-f name] [-h num] [-m num] [-n num] [-o list]
       [-p mode] [-r num] [-s num] [-v num] [-x num] [-y num] [file...]
```

```
/usr/lib/lp/postscript/postdaisy
```

### DESCRIPTION
The postdaisy filter translates Diablo 630 daisy-wheel files into PostScript and writes the results on the standard output. If no files are specified, or if `-` is one of the input files, the standard input is read.

### OPTIONS

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-c num</code></td>
<td>Print <code>num</code> copies of each page. By default only one copy is printed.</td>
</tr>
<tr>
<td><code>-f name</code></td>
<td>Print files using font <code>name</code>. Any PostScript font can be used, although the best results will be obtained only with constant-width fonts. The default font is Courier.</td>
</tr>
<tr>
<td><code>-h num</code></td>
<td>Set the initial horizontal motion index to <code>num</code>. Determines the character advance and the default point size, unless the <code>-s</code> option is used. The default is 12.</td>
</tr>
<tr>
<td><code>-m num</code></td>
<td>Magnify each logical page by the factor <code>num</code>. Pages are scaled uniformly about the origin, which is located near the upper left corner of each page. The default magnification is 1.0.</td>
</tr>
<tr>
<td><code>-n num</code></td>
<td>Print <code>num</code> logical pages on each piece of paper, where <code>num</code> can be any positive integer. By default, <code>num</code> is set to 1.</td>
</tr>
<tr>
<td><code>-o list</code></td>
<td>Print pages whose numbers are given in the comma-separated <code>list</code>. The list contains single numbers <code>N</code> and ranges <code>N1 – N2</code>. A missing <code>N1</code> means the lowest numbered page, a missing <code>N2</code> means the highest. The page range is an expression of logical pages rather than physical sheets of paper. For example, if you are printing two logical pages to a sheet, and you specified a range of 4, then two sheets of paper would print, containing four page layouts. If you specified a page range of 3-4, when requesting two logical pages to a sheet; then only page 3 and page 4 layouts would print, and they would appear on one physical sheet of paper.</td>
</tr>
<tr>
<td><code>-p mode</code></td>
<td>Print files in either portrait or landscape <code>mode</code>. Only the first character of <code>mode</code> is significant. The default <code>mode</code> is portrait.</td>
</tr>
<tr>
<td><code>-r num</code></td>
<td>Selects carriage return and line feed behavior. If <code>num</code> is 1, a line feed generates a carriage return. If <code>num</code> is 2, a carriage return generates a line feed. Setting <code>num</code> to 3 enables both modes.</td>
</tr>
<tr>
<td><code>-s num</code></td>
<td>Use point size <code>num</code> instead of the default value set by the initial horizontal motion index.</td>
</tr>
<tr>
<td><code>-v num</code></td>
<td>Set the initial vertical motion index to <code>num</code>. The default is 8.</td>
</tr>
<tr>
<td><code>-x num</code></td>
<td>Translate the origin <code>num</code> inches along the positive x axis. The default coordinate system has the origin fixed near the upper left corner of the...</td>
</tr>
</tbody>
</table>
postdaisy(1)

Page, with positive x to the right and positive y down the page. Positive
\textit{num} moves everything right. The default offset is \textit{0.25} inches.

\texttt{-y \textit{num}} Translate the origin \textit{num} inches along the positive y axis. Positive \textit{num}
moves text up the page. The default offset is \textit{−0.25} inches.

\textbf{EXIT STATUS}
The following exit values are returned:

\begin{itemize}
  \item \texttt{0} Successful completion.
  \item non-zero An error occurred.
\end{itemize}

\textbf{FILES}

\begin{itemize}
  \item \texttt{/usr/lib/lp/postscript/forms.ps}
  \item \texttt{/usr/lib/lp/postscript/ps.requests}
\end{itemize}

\textbf{ATTRIBUTES}

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

\begin{table}[h]
\begin{tabular}{|c|c|}
\hline
\textbf{ATTRIBUTE TYPE} & \textbf{ATTRIBUTE VALUE} \\
\hline
Availability & SUNWpsf \\
\hline
\end{tabular}
\end{table}

\textbf{SEE ALSO}
download(1), dpost(1), postdmd(1), postio(1), postmd(1), postprint(1),
postreverse(1), posttek(1), attributes(5)
postdmd – PostScript translator for DMD bitmap files

[-x num] [-y num] [file...]

/usr/lib/lp/postscript/postdmd

postdmd translates DMD bitmap files, as produced by \texttt{dmdps}, or files written in the Ninth Edition bitfile format into PostScript and writes the results on the standard output. If no files are specified, or if \texttt{-} is one of the input files, the standard input is read.

- \texttt{-b num} Pack the bitmap in the output file using \textit{num} byte patterns. A value of 0 turns off all packing of the output file. By default, \textit{num} is 6.

- \texttt{-c num} Print \textit{num} copies of each page. By default only one copy is printed.

- \texttt{-f} Flip the sense of the bits in files before printing the bitmaps.

- \texttt{-m num} Magnify each logical page by the factor \textit{num}. Pages are scaled uniformly about the origin, which by default is located at the center of each page. The default magnification is 1.0.

- \texttt{-n num} Print \textit{num} logical pages on each piece of paper, where \textit{num} can be any positive integer. By default \textit{num} is set to 1.

- \texttt{-o list} Print pages whose numbers are given in the comma-separated \textit{list}. The list contains single numbers \textit{N} and ranges \textit{N}1 \textendash \textit{N}2. A missing \textit{N}1 means the lowest numbered page, a missing \textit{N}2 means the highest. The page range is an expression of logical pages rather than physical sheets of paper. For example, if you are printing two logical pages to a sheet, and you specified a range of 4, then two sheets of paper would print, containing four page layouts. If you specified a page range of 3 \textendash 4, when requesting two logical pages to a sheet; then only page 3 and page 4 layouts would print, and they would appear on one physical sheet of paper.

- \texttt{-p mode} Print files in either portrait or landscape \textit{mode}. Only the first character of \textit{mode} is significant. The default \textit{mode} is portrait.

- \texttt{-x num} Translate the origin \textit{num} inches along the positive x axis. The default coordinate system has the origin fixed at the center of the page, with positive x to the right and positive y up the page. Positive \textit{num} moves everything right. The default offset is 0 inches.

- \texttt{-y num} Translate the origin \textit{num} inches along the positive y axis. Positive \textit{num} moves everything up the page. The default offset is 0.

Only one bitmap is printed on each logical page, and each of the input files must contain complete descriptions of at least one bitmap. Decreasing the pattern size using the \texttt{-b} option may help throughput on printers with fast processors (such as PS-810s), while increasing the pattern size will often be the right move on older models (such as PS-800s).
postmd(1)

EXIT STATUS
The following exit values are returned:
0 Successful completion.
non-zero An error occurred.

FILES
/usr/lib/lp/postscript/forms.ps
/usr/lib/lp/postscript/ps.requests

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWpsf</td>
</tr>
</tbody>
</table>

SEE ALSO
download(1), dpost(1), postdaisy(1), postio(1), postmd(1), postprint(1), postreverse(1), posttek(1), attributes(5)
NAME
postio – serial interface for PostScript printers

SYNOPSIS
[-P string] [-R num] [file...]

/usr/lib/lp/postscript/postio

DESCRIPTION
postio sends files to the PostScript printer attached to line. If no files are specified the
standard input is sent.

OPTIONS
The first group of options should be sufficient for most applications:
- D
  Enable debug mode. Guarantees that everything read on line will
  be added to the log file (standard error by default).
- q
  Prevents status queries while files are being sent to the printer.
  When status queries are disabled a dummy message is appended
  to the log file before each block is transmitted.
- b speed
  Transmit data over line at baud rate speed. Recognized baud rates
  are 1200, 2400, 4800, 9600, and 19200. The default speed is 9600
  baud.
- B num
  Set the internal buffer size for reading and writing files to num
  bytes. By default num is 2048 bytes.
- l line
  Connect to the printer attached to line. In most cases there is no
  default and postio must be able to read and write line. If the
  line does not begin with a / it may be treated as a Datakit
destination.
- L file
  Data received on line gets put in file. The default log file is
  standard error. Printer or status messages that don't show a
  change in state are not normally written to file but can be forced
  out using the -D option.
- P string
  Send string to the printer before any of the input files. The default
  string is simple PostScript code that disables timeouts.
- R num
  Run postio as a single process if num is 1 or as separate read and
  write processes if num is 2. By default postio runs as a single
  process.

The next two options are provided for users who expect to run postio on their own.
Neither is suitable for use in spooler interface programs:
- i
  Run the program in interactive mode. Any files are sent first and followed
  by the standard input. Forces separate read and write processes and
  overrides many other options. To exit interactive mode use your interrupt
  or quit character. To get a friendly interactive connection with the printer
type executive on a line by itself.
Data received on line and not recognized as printer or status information is written to the standard output. Forces separate read and write processes. Convenient if you have a PostScript program that will be returning useful data to the host.

The last option is not generally recommended and should only be used if all else fails to provide a reliable connection:

-S Slow the transmission of data to the printer. Severely limits throughput, runs as a single process, disables the -q option, limits the internal buffer size to 1024 bytes, can use an excessive amount of CPU time, and does nothing in interactive mode.

The best performance will usually be obtained by using a large internal buffer (the -B option) and by running the program as separate read and write processes (the -R 2 option). Inability to fork the additional process causes postio to continue as a single read/write process. When one process is used, only data sent to the printer is flow controlled.

The options are not all mutually exclusive. The -i option always wins, selecting its own settings for whatever is needed to run interactive mode, independent of anything else found on the command line. Interactive mode runs as separate read and write processes and few of the other options accomplish anything in the presence of the -i option. The -t option needs a reliable two way connection to the printer and therefore tries to force separate read and write processes. The -S option relies on the status query mechanism, so -q is disabled and the program runs as a single process.

In most cases postio starts by making a connection to line and then attempts to force the printer into the IDLE state by sending an appropriate sequence of ^T (status query), ^C (interrupt), and ^D (end of job) characters. When the printer goes IDLE, files are transmitted along with an occasional ^T (unless the -q option was used). After all the files are sent the program waits until it’s reasonably sure the job is complete. Printer generated error messages received at any time except while establishing the initial connection (or when running interactive mode) cause postio to exit with a non-zero status. In addition to being added to the log file, printer error messages are also echoed to standard error.

EXAMPLES

**EXAMPLE 1** Examples of the postio command.

Run as a single process at 9600 baud and send file1 and file2 to the printer attached to /dev/tty01:

```
example% postio -l /dev/tty01 file1 file2
```

Same as above except two processes are used, the internal buffer is set to 4096 bytes, and data returned by the printer gets put in file log:

```
example% postio -R 2 -B 4096 -l/dev/tty01 -L file1 file2
```

Establish an interactive connection with the printer at Datakit destination my/printer:
EXAMPLE 1 Examples of the `postio` command.  (Continued)

```bash
example$ postio -i -l my/printer
```

Send file program to the printer connected to `/dev/tty22`, recover any data in file results, and put log messages in file `log`:

```bash
example$ postio -t -l /dev/tty22 -L log program >results
```

EXIT STATUS The following exit values are returned:

- **0**  Successful completion.
- **non-zero**  An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWpsf</td>
</tr>
</tbody>
</table>

SEE ALSO `download(1)`, `dpost(1)`, `postdaisy(1)`, `postmd(1)`, `postprint(1)`, `postreverse(1)`, `posttek(1)`, `attributes(5)`

NOTES The input files are handled as a single PostScript job. Sending several different jobs, each with their own internal end of job mark (`^D`) is not guaranteed to work properly. `postio` may quit before all the jobs have completed and could be restarted before the last one finishes.

All the capabilities described above may not be available on every machine or even across the different versions of the UNIX system that are currently supported by the program.

There may be no default line, so using the `-l` option is strongly recommended. If omitted, `postio` may attempt to connect to the printer using the standard output. If Datakit is involved, the `-b` option may be ineffective and attempts by `postio` to impose flow control over data in both directions may not work. The `-q` option can help if the printer is connected to RADIUS. The `-S` option is not generally recommended and should be used only if all other attempts to establish a reliable connection fail.
NAME
postmd – matrix display program for PostScript printers

SYNOPSIS
postmd [-b num] [-c num] [-d dimen] [-g list] [-i list] [-m num]
[-n num] [-o list] [-p mode] [-w window] [-x num] [-y num] [file...]
/usr/lib/lp/postscript/postmd

DESCRIPTION
The postmd filter reads a series of floating point numbers from files, translates them into a PostScript gray scale image, and writes the results on the standard output. In a typical application, the numbers might be the elements of a large matrix, written in row major order, while the printed image could help locate patterns in the matrix. If no files are specified, or if - is one of the input files, the standard input is read.

OPTIONS
The following options are supported:

- `-b num` Packs the bitmap in the output file using num byte patterns. A value of 0 turns off all packing of the output file. By default, num is 6.

- `-c num` Prints num copies of each page. By default, only one copy is printed.

- `-d dimen` Sets the default matrix dimensions for all input files to dimen. The dimen string can be given as rows or rowsxcolumns. If columns is omitted it will be set to rows. By default, postmd assumes each matrix is square and sets the number of rows and columns to the square root of the number of elements in each input file.

- `-g list` list is a comma- or space-separated string of integers, each lying between 0 and 255 inclusive, that assigns PostScript gray scales to the regions of the real line selected by the -i option. 255 corresponds to white, and 0, to black. The postmd filter assigns a default gray scale that omits white (that is, 255) and gets darker as the regions move from left to right along the real line.

- `-i list` list is a comma-, space-, or slash(/)-separated string of N floating point numbers that partition the real line into 2N+1 regions. The list must be given in increasing numerical order. The partitions are used to map floating point numbers read from the input files into gray scale integers that are either assigned automatically by postmd or arbitrarily selected using the -g option. The default interval list is -1, 0, 1, which partitions the real line into seven regions.

- `-m num` Magnifies each logical page by the factor num. Pages are scaled uniformly about the origin which, by default, is located at the center of each page. The default magnification is 1.0.

- `-n num` Prints num logical pages on each piece of paper, where num can be any positive integer. By default, num is set to 1.
Prints pages whose numbers are given in the comma separated list. The list contains single numbers \( N \) and ranges \( N1 - N2 \). A missing \( N1 \) means the lowest numbered page, a missing \( N2 \) means the highest. The page range is an expression of logical pages rather than physical sheets of paper. For example, if you are printing two logical pages to a sheet, and you specified a range of 4, then two sheets of paper would print, containing four page layouts. If you specified a page range of 3-4, when requesting two logical pages to a sheet; then only page 3 and page 4 layouts would print, and they would appear on one physical sheet of paper.

Prints files in either portrait or landscape mode. Only the first character of mode is significant. The default mode is portrait.

\( window \) is a comma- or space-separated list of four positive integers that select the upper left and lower right corners of a submatrix from each of the input files. Row and column indices start at 1 in the upper left corner and the numbers in the input files are assumed to be written in row major order. By default, the entire matrix is displayed.

Translates the origin \( num \) inches along the positive x axis. The default coordinate system has the origin fixed at the center of the page, with positive x to the right and positive y up the page. Positive \( num \) moves everything right. The default offset is 0 inches.

Translates the origin \( num \) inches along the positive y axis. Positive \( num \) moves everything up the page. The default offset is 0.

Only one matrix is displayed on each logical page, and each of the input files must contain complete descriptions of exactly one matrix. Matrix elements are floating point numbers arranged in row major order in each input file. White space, including newlines, is not used to determine matrix dimensions. By default, \texttt{postmd} assumes each matrix is square and sets the number of rows and columns to the square root of the number of elements in the input file. Supplying default dimensions on the command line with the \(-d\) option overrides this default behavior, and in that case the dimensions apply to all input files.

An optional header can be supplied with each input file and is used to set the matrix dimensions, the partition of the real line, the gray scale map, and a window into the matrix. The header consists of keyword/value pairs, each on a separate line. It begins on the first line of each input file and ends with the first unrecognized string, which should be the first matrix element. Values set in the header take precedence, but apply only to the current input file. Recognized header keywords are \texttt{dimension}, \texttt{interval}, \texttt{grayscale}, and \texttt{window}. The syntax of the value string that follows each keyword parallels what is accepted by the \(-d\), \(-i\), \(-g\), and \(-w\) options.
EXAMPLES

EXAMPLE 1 Generating an interval list

For example, suppose file initially contains the 1000 numbers in a 20x50 matrix. Then you can produce exactly the same output by completing three steps.

1. First, issue the following command line:
   example% postmd -d20x50 -i"-100 100" -g0,128,254,128,0 file

2. Second, prepend the following header to file:
   example% postmd -d20x50 -i"-100 100" -g0,128,254,128,0 file

3. Third, issue the following command line:
   example% postmd file

The interval list partitions the real line into five regions and the gray scale list maps numbers less than –100 or greater than 100 into 0 (that is, black), numbers equal to –100 or 100 into 128 (that is, 50 percent black), and numbers between –100 and 100 into 254 (that is, almost white).

FILES
/usr/lib/lp/postscript/forms.ps
/usr/lib/lp/postscript/ps.requests

EXIT STATUS

The following exit values are returned:

0 Successful completion.
non-zero An error occurred.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWpsf</td>
</tr>
</tbody>
</table>

SEE ALSO

dpost(1), postdaisy(1), postmd(1), postio(1), postprint(1), postreverse(1), posttek(1), attributes(5)

NOTES

The largest matrix that can be adequately displayed is a function of the interval and gray scale lists, the printer resolution, and the paper size. A 600 by 600 matrix is an optimistic upper bound for a two element interval list (that is, five regions) using 8.5 by 11 inch paper on a 300 dpi printer.

Using white (that is, 255) in a gray scale list is not recommended and won’t show up in the legend and bar graph that postmd displays below each image.
postplot – PostScript translator for plot(4B) graphics files

SYNOPSIS
postplot [-c num] [-f name] [-m num] [-n num] [-o list] [-p mode]
[-w num] [-x num] [-y num] [filename...]

/usr/lib/lp/postscript/postplot

DESCRIPTION
The postplot filter translates plot(1B) graphics filenames into PostScript and writes the results on the standard output. If no filenames are specified, or if – is one of the input filenames, the standard input is read.

OPTIONS
The following options are supported:

-c num  Print num copies of each page. By default, only one copy is printed.

-f name  Print text using font name. Any PostScript font can be used, although the best results will be obtained only with constant width fonts. The default font is Courier.

-m num  Magnify each logical page by the factor num. Pages are scaled uniformly about the origin which, by default, is located at the center of each page. The default magnification is 1.0.

-n num  Print num logical pages on each piece of paper, where num can be any positive integer. By default, num is set to 1.

-o list  Print pages whose numbers are given in the comma-separated list. The list contains single numbers \( N \) and ranges \( N_1 - N_2 \). A missing \( N_1 \) means the lowest numbered page, a missing \( N_2 \) means the highest.

-p mode  Print filenames in either portrait or landscape mode. Only the first character of mode is significant. The default mode is landscape.

-w num  Set the line width used for graphics to num points, where a point is approximately 1/72 of an inch. By default, num is set to 0 points, which forces lines to be one pixel wide.

-x num  Translate the origin num inches along the positive x axis. The default coordinate system has the origin fixed at the center of the page, with positive x to the right and positive y up the page. Positive num moves everything right. The default offset is 0.0 inches.

-y num  Translate the origin num inches along the positive y axis. Positive num moves everything up the page. The default offset is 0.0.

OPERANDS
The following operand is supported:

filename  The graphics filename to be translated

EXIT STATUS
The following exit value is returned:

0  filename(s) were successfully processed.
ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWlps</td>
</tr>
</tbody>
</table>

SEE ALSO
download(1), dpost(1), plot(1B), postdaisy(1), postmd(1), postio(1), postmd(1), postprint(1), postreverse(1), plot(4B), attributes(5)

NOTES

The default line width is too small for write-white print engines, such as the one used by the PS-2400.
### postprint(1)

<table>
<thead>
<tr>
<th><strong>NAME</strong></th>
<th>postprint – PostScript translator for text files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DESCRIPTION</strong></td>
<td>The postprint filter translates text files into PostScript and writes the results on the standard output. If no files are specified, or if – is one of the input files, the standard input is read.</td>
</tr>
</tbody>
</table>
| **OPTIONS** | -c num: Print num copies of each page. By default, only one copy is printed.  
- f name: Print files using font name. Any PostScript font can be used, although the best results will be obtained only with constant width fonts. The default font is Courier.  
-l num: Set the length of a page to num lines. By default, num is 66. Setting num to 0 is allowed, and will cause postprint to guess a value, based on the point size that’s being used.  
-m num: Magnify each logical page by the factor num. Pages are scaled uniformly about the origin, which is located near the upper left corner of each page. The default magnification is 1.0.  
-n num: Print num logical pages on each piece of paper, where num can be any positive integer. By default, num is set to 1.  
-o list: Print pages whose numbers are given in the comma-separated list. The list contains single numbers N and ranges N1 – N2. A missing N1 means the lowest numbered page, a missing N2 means the highest. The page range is an expression of logical pages rather than physical sheets of paper. For example, if you are printing two logical pages to a sheet, and you specified a range of 4, then two sheets of paper would print, containing four page layouts. If you specified a range of 3–4, when requesting two logical pages to a sheet; then only page 3 and page 4 layouts would print, and they would appear on one physical sheet of paper.  
-p mode: Print files in either portrait or landscape mode. Only the first character of mode is significant. The default mode is portrait.  
-r num: Selects carriage return behavior. Carriage returns are ignored if num is 0, cause a return to column 1 if num is 1, and generate a newline if num is 2. The default num is 0.  
-s num: Print files using point size num. When printing in landscape mode num is scaled by a factor that depends on the imaging area of the device. The default size for portrait mode is 10. Note that increasing point size increases virtual image size, so you either need to load larger paper, or use the −10 option to scale the number of lines per page.  
-t num: Assume tabs are set every num columns, starting with the first column. By default, tabs are set every 8 columns. |

User Commands  1155
postprint(1)

-x num  Translate the origin num inches along the positive x axis. The default coordinate system has the origin fixed near the upper left corner of the page, with positive x to the right and positive y down the page. Positive num moves everything to the right. The default offset is 0.25 inches.

-y num  Translate the origin num inches along the positive y axis. Positive num moves text up the page. The default offset is −0.25 inches.

A new logical page is started after 66 lines have been printed on the current page, or whenever an ASCII form feed character is read. The number of lines per page can be changed using the -l option. Unprintable ASCII characters are ignored, and lines that are too long are silently truncated by the printer.

EXAMPLES  EXAMPLE 1 Examples of postprint.

To print file1 and file2 in landscape mode, issue the following command:

```bash
example$ postprint -pland file1 file2
```

To print three logical pages on each physical page in portrait mode:

```bash
example$ postprint -n3 file
```

EXIT STATUS  The following exit values are returned:

0  Successful completion.
non-zero  An error occurred.

FILES /usr/lib/lp/postscript/forms.ps
/usr/lib/lp/postscript/ps.requests

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWpsf</td>
</tr>
</tbody>
</table>

SEE ALSO  download(1), dpost(1), postdaisy(1), postdmd(1), postio(1), postmd(1), postreverse(1), posttek(1), attributes(5)
postreverse – reverse the page order in a PostScript file

**SYNOPSIS**

postreverse [-o list] [-r] [file]

/usr/lib/lp/postscript/postreverse

**DESCRIPTION**

The `postreverse` filter reverses the page order in files that conform to Adobe's Version 1.0 or Version 2.0 file structuring conventions, and writes the results on the standard output. Only one input file is allowed and if no file is specified, the standard input is read.

The `postreverse` filter can handle a limited class of files that violate page independence, provided all global definitions are bracketed by `%%BeginGlobal` and `%%EndGlobal` comments. In addition, files that mark the end of each page with `%%EndPage: label ordinal` comments will also reverse properly, provided the prologue and trailer sections can be located. If `postreverse` fails to find an `%%EndProlog` or `%%EndSetup` comment, the entire file is copied, unmodified, to the standard output.

Because global definitions are extracted from individual pages and put in the prologue, the output file can be minimally conforming, even if the input file was not.

**OPTIONS**

- `-o list` Select pages whose numbers are given in the comma-separated list. The list contains single numbers N and ranges N1 – N2. A missing N1 means the lowest numbered page, a missing N2 means the highest. The page range is an expression of logical pages rather than physical sheets of paper. For example, if you are printing two logical pages to a sheet, and you specified a range of 4, then two sheets of paper would print, containing four page layouts. If you specified a page range of 3-4, when requesting two logical pages to a sheet; then only page 3 and page 4 layouts would print, and they would appear on one physical sheet of paper.

- `-r` Do not reverse the pages in file.

**EXAMPLES**

**EXAMPLE 1** Examples of `postreverse`.

To select pages 1 to 100 from file and reverse the pages:

```
example$ postreverse -o1-100 file
```

To print four logical pages on each physical page and reverse all the pages:

```
example$ postprint -n4 file | postreverse
```

To produce a minimally conforming file from output generated by `dpost` without reversing the pages:

```
example$ dpost file | postreverse -r
```

**EXIT STATUS**

The following exit values are returned:

0 Successful completion.
postreverse(1)

non-zero

An error occurred.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWpsf</td>
</tr>
</tbody>
</table>

SEE ALSO
download(1), dpost(1), postdaisy(1), postmd(1), postio(1), postmd(1), postprint(1), posttek(1), attributes(5)

NOTES

No attempt has been made to deal with redefinitions of global variables or procedures. If standard input is used, the input file will be read three times before being reversed.
NAME  posttek – PostScript translator for Tektronix 4014 files

SYNOPSIS  posttek [-c num] [-f name] [-m num] [-n num] [-o list] [-p mode]
            [-w num] [-x num] [-y num] [file...]
/usr/lib/lp/postscript/posttek

DESCRIPTION  The posttek filter translates Tektronix 4014 graphics files into PostScript and writes
the results on the standard output. If no files are specified, or if – is one of the input
files, the standard input is read.

OPTIONS  
- c num  Print num copies of each page. By default, only one copy is printed.
- f name  Print text using font name. Any PostScript font can be used, although the
best results will be obtained only with constant width fonts. The default
font is Courier.
- m num  Magnify each logical page by the factor num. Pages are scaled uniformly
about the origin which, by default, is located at the center of each page. The
default magnification is 1.0.
- n num  Print num logical pages on each piece of paper, where num can be any
positive integer. By default, num is set to 1.
- o list  Print pages whose numbers are given in the comma-separated list. The list
contains single numbers N and ranges N1 – N2. A missing N1 means the
lowest numbered page, a missing N2 means the highest. The page range is
an expression of logical pages rather than physical sheets of paper. For
example, if you are printing two logical pages to a sheet, and you specified
a range of 4, then two sheets of paper would print, containing four page
layouts. If you specified a page range of 3-4, when requesting two logical
pages to a sheet; then only page 3 and page 4 layouts would print, and they
would appear on one physical sheet of paper.
- p mode  Print files in either portrait or landscape mode. Only the first character of
mode is significant. The default mode is landscape.
- w num  Set the line width used for graphics to num points, where a point is
approximately 1/72 of an inch. By default, num is set to 0 points, which
forces lines to be one pixel wide.
- x num  Translate the origin num inches along the positive x axis. The default
coordinate system has the origin fixed at the center of the page, with
positive x to the right and positive y up the page. Positive num moves
everything right. The default offset is 0.0 inches.
- y num  Translate the origin num inches along the positive y axis. Positive num
moves everything up the page. The default offset is 0.0.

EXIT STATUS  The following exit values are returned:
0  Successful completion.
non-zero  An error occurred.
FILEs

/usr/lib/lp/postscript/forms.ps
/usr/lib/lp/postscript/ps.requests

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWpsf</td>
</tr>
</tbody>
</table>

SEE ALSO
download(1), dpost(1), postdaisy(1), postdmd(1), postio(1), postmd(1),
postprint(1), postreverse(1), attributes(5)

NOTES
The default line width is too small for write-white print engines, such as the one used
by the PS-2400.
NAME

ppgsz – set preferred stack and/or heap page size

SYNOPSIS

/usr/bin/ppgsz [-F] -o option [, option] cmd | -p pid...

DESCRIPTION

The ppgsz utility sets the preferred stack and/or heap page size for the target process(es), that is, the launched cmd or the process(es) in the pid list. ppgsz stops the target process(es) while changing the page size. See memcntl(2).

OPTIONS

The following options are supported:

- F

Force. Sets the preferred page size options(s) for target process(es) even if controlled by other process(es). Caution should be exercised when using the - F flag. See proc(1).

- p pid

Sets the preferred page size option(s) for the target process(es) in the process-id (pid) list following the - p option. The pid list can also consist of names in the /proc directory. Only the process owner or the super-user is permitted to set page size.

cmd is interpreted if -p is not specified. ppgsz launches cmd and applies page size option(s) to the new process.

The heap and stack preferred page sizes are inherited. Child process(es) created (see fork(2)) from the launched process or the target process(es) in the pid list after ppgsz completes will inherit the preferred heap and stack page sizes. The preferred page sizes are set back to the default system page size on exec(2) (see getpagesize(3C)).

- o option [, option]

The options are:

heap=size

This option specifies the preferred page size for the heap of the target process(es). heap is defined to be the bss (uninitialized data) and the brk area that immediately follows the bss (see brk(2)). The preferred heap page size is set for the existing heap and for any additional heap memory allocated in the future. See NOTES.

stack=size

This option specifies the preferred page size for the stack of the target process(es). The preferred stack page size is set for the existing stack and newly allocated parts of the stack as it expands.

At least one of the above options must be specified.

size must be a supported page size (see pagesize(1)) or 0, in which case the system will select an appropriate page size (see memcntl(2)).
size defaults to bytes and can be specified in octal (0), decimal, or hexadecimal (0x). The numeric value can be qualified with K, M, G, or T to specify Kilobytes, Megabytes, Gigabytes, or Terabytes, respectively. 4194304, 0x400000, 4096K, 0x1000K, and 4M are different ways to specify 4 Megabytes.

EXAMPLE 1 Setting the preferred heap and stack page size

The following example sets the preferred heap page size to 4M and the preferred stack page size to 512K for all ora—owned processes running commands that begin with ora:

eEXAMPLE% ppgsz -o heap=4M,stack=512K -p 'pgrep -u ora "ora"'

EXIT STATUS If cmd is specified and successfully invoked (see exec(2)), the exit status of ppgsz will be the exit status of cmd. Otherwise, ppgsz will exit with one of the following values:

<table>
<thead>
<tr>
<th>Exit Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successfully set preferred page size(s) for processes in the pid list.</td>
</tr>
<tr>
<td>125</td>
<td>An error occurred in ppgsz. Errors include: invalid argument, invalid page size(s) specified, and failure to set preferred page size(s) for one or more processes in the pid list or cmd.</td>
</tr>
<tr>
<td>126</td>
<td>cmd was found but could not be invoked.</td>
</tr>
<tr>
<td>127</td>
<td>cmd could not be found.</td>
</tr>
</tbody>
</table>

FILES

/proc/* Process files.
/usr/lib/ld/map.bssalign A template link-editor mapfile for aligning bss (see NOTES).

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>Attribute Type</th>
<th>Attribute Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu (32-bit)</td>
</tr>
<tr>
<td></td>
<td>SUNWesxu (64-bit)</td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Evolving</td>
</tr>
</tbody>
</table>

SEE ALSO

ld(1), mpss.so.1(1), pagesize(1), pgrep(1), pmap(1), proc(1), brk(2), exec(2), fork(2), memcntl(2), sbrk(2), getpagesize(3C), proc(4), attributes(5)

Linker and Libraries Guide

NOTES

Due to resource constraints, the setting of the preferred page size does not necessarily guarantee that the target process(es) will get the preferred page size. Use pmap(1) to view the actual heap and stack page sizes of the target process(es) (see pmap -s option).
Large pages are required to be mapped at addresses that are multiples of the size of the large page. Given that the heap is typically not large page aligned, the starting portions of the heap (below the first large page aligned address) are mapped with the system memory page size. See `getpagesize(3C)`.

To provide a heap that will be mapped with a large page size, an application can be built using a link-editor (`ld(1)` `mapfile` containing the `bss` segment declaration directive. Refer to the section “Mapfile Option” in the Linker and Libraries Guide for more details of this directive and the template `mapfile` provided in `/usr/lib/ld/map.bssalign`. Users are cautioned that an alignment specification may be machine-specific and may lose its benefit on different hardware platforms. A more flexible means of requesting the most optimal underlying page size may evolve in future releases.

`mpss.so.1(1)`, a preloadable shared object, can also be used to set the preferred stack and/or heap page sizes.
pr(1)

NAME  pr - print files

SYNOPSIS  /usr/bin/pr [ + page] [-column] [-adFmrt] [-e [char] [gap]] [-h header]
            [-l [char] [gap]] [-l lines] [-n [char] [width]] [-o offset] [-s [char]] [-w width] [-fp] [file...]

/usr/xpg4/bin/pr [ + page] [-column] [-c column] [-adFmrt] [-e [char] [gap]] [-h header] [-i [char] [gap]] [-l lines] [-n [char] [width]]
            [-o offset] [-s [char]] [-w width] [-fp] [file...]

DESCRIPTION  The pr utility is a printing and pagination filter. If multiple input files are specified,
each is read, formatted, and written to standard output. By default, the input is
separated into 66-line pages, each with:

- a 5-line header that includes the page number, date, time and the path name of the
  file
- a 5-line trailer consisting of blank lines

If standard output is associated with a terminal, diagnostic messages will be deferred
until the pr utility has completed processing.

When options specifying multi-column output are specified, output text columns will
be of equal width; input lines that do not fit into a text column will be truncated. By
default, text columns are separated with at least one blank character.

OPTIONS  The following options are supported. In the following option descriptions, column,
lines, offset, page, and width are positive decimal integers; gap is a non-negative
decimal integer. Some of the option-arguments are optional, and some of the
option-arguments cannot be specified as separate arguments from the preceding
option letter. In particular, the -s option does not allow the option letter to be
separated from its argument, and the options -e, -i, and -n require that both
arguments, if present, not be separated from the option letter.

The following options are supported for both /usr/bin/pr and
/usr/xpg4/bin/pr:

+page  Begin output at page number page of the formatted input.

-column  Produce multi-column output that is arranged in
column columns (default is 1) and is written down each
column in the order in which the text is received from
the input file. This option should not be used with -m.
The -e and -i options will be assumed for multiple
text-column output. Whether or not text columns are
produced with identical vertical lengths is unspecified,
but a text column will never exceed the length of the
page (see the -l option). When used with -t, use the
minimum number of lines to write the output.
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-a</code></td>
<td>Modify the effect of the <code>-column</code> option so that the columns are filled across the page in a round-robin order (for example, when <code>column</code> is 2, the first input line heads column 1, the second heads column 2, the third is the second line in column 1, and so forth).</td>
</tr>
<tr>
<td><code>-d</code></td>
<td>Produce output that is double-spaced; append an extra <code>NEWLINE</code> character following every <code>NEWLINE</code> character found in the input.</td>
</tr>
<tr>
<td><code>-e [ char ] [ gap ]</code></td>
<td>Expand each input TAB character to the next greater column position specified by the formula <code>n * gap + 1</code>, where <code>n</code> is an integer &gt; 0. If <code>gap</code> is 0 or is omitted, it defaults to 8. All TAB characters in the input will be expanded into the appropriate number of SPACE characters. If any non-digit character, <code>char</code>, is specified, it will be used as the input tab character.</td>
</tr>
<tr>
<td><code>-f</code></td>
<td>Use a FORMFEED character for new pages, instead of the default behavior that uses a sequence of <code>NEWLINE</code> characters. Pause before beginning the first page if the standard output is associated with a terminal.</td>
</tr>
<tr>
<td><code>-h header</code></td>
<td>Use the string <code>header</code> to replace the contents of the <code>file</code> operand in the page header.</td>
</tr>
<tr>
<td><code>-l lines</code></td>
<td>Override the 66-line default and reset the page length to <code>lines</code>. If <code>lines</code> is not greater than the sum of both the header and trailer depths (in lines), <code>pr</code> will suppress both the header and trailer, as if the <code>-t</code> option were in effect.</td>
</tr>
<tr>
<td><code>-m</code></td>
<td>Merge files. Standard output will be formatted so <code>pr</code> writes one line from each file specified by <code>file</code>, side by side into text columns of equal fixed widths, in terms of the number of column positions. Implementations support merging of at least nine files.</td>
</tr>
<tr>
<td><code>-n [ char ] [ width ]</code></td>
<td>Provide <code>width</code>-digit line numbering (default for <code>width</code> is 5). The number will occupy the first <code>width</code> column positions of each text column of default output or each line of <code>-m</code> output. If <code>char</code> (any non-digit character) is given, it will be appended to the line number to separate it from whatever follows (default for <code>char</code> is a TAB character).</td>
</tr>
<tr>
<td><code>-o offset</code></td>
<td>Each line of output will be preceded by <code>offset</code> &lt;space&gt;s. If the <code>-o</code> option is not specified, the default offset is 0. The space taken will be in addition to the output line width (see <code>-w</code> option below).</td>
</tr>
</tbody>
</table>
pr(1)

-p     Pause before beginning each page if the standard output is directed to a terminal (`pr` will write an ALERT character to standard error and wait for a carriage-return character to be read on `/dev/tty`).

-r     Write no diagnostic reports on failure to open files.

-s [ char ]     Separate text columns by the single character `char` instead of by the appropriate number of SPACE characters (default for `char` is the TAB character).

-t     Write neither the five-line identifying header nor the five-line trailer usually supplied for each page. Quit writing after the last line of each file without spacing to the end of the page.

-w width     Set the width of the line to `width` column positions for multiple text-column output only. If the `-w` option is not specified and the `-s` option is not specified, the default width is 72. If the `-w` option is not specified and the `-s` option is specified, the default width is 512.

For single column output, input lines will not be truncated.

/usr/bin/pr   The following options are supported for `/usr/bin/pr` only:

-F     Fold the lines of the input file. When used in multi-column mode (with the `-a` or `-m` options), lines will be folded to fit the current column's width; otherwise, they will be folded to fit the current line width (80 columns).

-i [ char ] [ gap ]     output, replace SPACE characters with TAB characters wherever one or more adjacent SPACE characters reach column positions `gap+1`, `2*gap+1`, `3*gap+1`, and so forth. If `gap` is 0 or is omitted, default TAB settings at every eighth column position are assumed. If any non-digit character, `char`, is specified, it will be used as the output TAB character.

/usr/xpg4/bin/pr  The following options are supported for `/usr/xpg4/bin/pr` only:

-F     Use a FORMFEED character for new pages, instead of the default behavior that uses a sequence of NEWLINE characters.

-i [ char ] [ gap ]     output, replace multiple SPACE characters with TAB characters wherever two or more adjacent SPACE characters reach column positions `gap+1`, `2*gap+1`, `3*gap+1`, and so forth. If `gap` is 0 or is omitted, default TAB settings at every eighth column position are assumed. If any non-digit character, `char`, is specified, it will be used as the output TAB character.

OPERANDS   The following operand is supported:

1166 man pages section 1: User Commands • Last Revised 18 Mar 1997
file  A path name of a file to be written. If no file operands are specified, or if a file operand is —, the standard input will be used.

EXAMPLES

EXAMPLE 1 Print a numbered list of all files in the current directory:

```
ls -a | pr -n -h "Files in $(pwd)."
```

EXAMPLE 2 Print file1 and file2 as a double-spaced, three-column listing headed by “file list”:

```
pr -3d -h "file list" file1 file2
```

EXAMPLE 3 Write file1 on file2, expanding tabs to columns 10, 19, 28, ...:

```
pr -e9 -t <file1 >file2
```

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of pr: LC_CTYPE, LC_MESSAGES, LC_TIME, TZ, and NLSPATH.

EXIT STATUS

The following exit values are returned:

- 0  Successful completion.
- >0  An error occurred.

ATTRIBUTES

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

```
ATTRIBUTES
<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>
```

SEE ALSO

expand(1), lp(1), attributes(5), environ(5), XPG4(5)
praliases(1)

NAME  praliases – display system mail aliases

SYNOPSIS  praliases [-c configfile] [-f aliasfile] [key]

DESCRIPTION  The praliases utility displays system mail aliases. When no key is given, praliases displays the current system aliases, one per line, in no particular order. The form is key:value. If a key is given, only that key is looked up and the appropriate key:value is displayed if found.

OPTIONS  The following options are supported:

- -c configfile  Specifies a sendmail configuration file.
- -f aliasfile  Reads the specified file aliasfile instead of the default sendmail system aliases file.

OPERANDS  The following operands are supported:

key  A specific alias key to look up.

EXIT STATUS  The following exit values are returned:

0  Successful operation.
>0  An error occurred.

FILES  /etc/mail/aliases  Default sendmail system aliases file
/etc/mail/aliases.db  Database versions of the
/etc/mail/aliases file
/etc/mail/aliases.dir
/etc/mail/aliases.pag  Database versions of the
/etc/mail/aliases file
/etc/mail/sendmail.cf  Default sendmail configuration file

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWandmu</td>
</tr>
</tbody>
</table>

SEE ALSO  mailq(1), newaliases(1M), sendmail(1M), attributes(5)
NAME  prctl – get or set the resource controls of running processes, tasks, and projects

SYNOPSIS  prctl [-t [basic | privileged | system]] [-e | -d action] [-rx] [-n name [-v value]] [-i idtype] [id...]

DESCRIPTION  The prctl utility allows the examination and modification of the resource controls associated with an active process, task, or project on the system. It allows access to the basic and privileged limits on the specified entity.

OPTIONS  The following options are supported:

-d | -e action  Enables (-e) or disables (-d) the specified action on the specified resource control. The special token all is valid with the disable option to deactivate all actions on the given resource control value.

The other defined actions for a resource are deny and signal=signum. The deny action indicates that the resource control encountered will attempt to deny granting the resource to the process, task, or project on a request for resources in excess of the value provided by the -v option for the new resource control. In the signal=signum action, signum is a signal number (or string representation of a signal). deny actions may not be activated or deactivated if global flags indicate that the deny action is unchangeable.

-i idtype  Specifies the type of the id operands. Valid idtypes are process, task, or project. The default id type, if the -i option is omitted, is process.

-n name  Specifies the name of the resource control to get or set. If the name is unspecified, all resource controls are retrieved.

-r  Replaces the first resource control value (matching with the -t privilege) with the new value specified through the -v option.

-t [basic | privileged | system]  Specifies which resource control type to set. Unless the "lowerable" flag is set for a resource control, only invocations by users (or setuid programs) who have privileges equivalent to those of root can modify privileged resource controls. See
prctl(1)

prctl(1)

rctlblk_set_value(3C) for a description of the RCTL_GLOBAL_LOWERABLE flag. If the type is not specified, basic is assumed. For a get operation, the values of all resource control types, including system, are displayed if no type is specified.

- v value

Specifies the value for the resource control for a set operation. If no value is specified, then the modification (deletion, action enabling or disabling) will be carried out on the lowest-valued resource control with the given type.

- x

Deletes the specified resource control value. If the delete option is not provided, the default operation of prctl is to modify a resource control value of matching value and privilege, or insert a new value with the given privilege. The matching criteria are discussed more fully in setrctl(2).

If none of the -d, -e, -v, or -x options is specified, the invocation is considered a get operation.

OPERANDS

The following operand is supported:

id

The ID of the entity (process, task, or project) to interrogate. If the invoking user's credentials are unprivileged and the entity being interrogated possesses different credentials, the operation will fail. If no id is specified, an error message is returned.

EXAMPLES

EXAMPLE 1 Displaying current resource control settings for a specific process

example$ pgrep sort
111759
example$ prctl 111759
111759: /usr/bin/sort
process.max-address-space [ lowerable deny no-local-action ]
  184467440709551615 privileged deny
  184467440709551615 system deny
process.max-file-descriptor [ lowerable deny ]
  256 basic deny
  65536 privileged deny
  2147483647 system deny
process.max-core-size [ lowerable deny no-local-action ]
  184467440709551615 privileged deny
  184467440709551615 system deny
process.max-stack-size [ lowerable deny no-local-action ]
  8388608 basic deny
  9223372036854775807 privileged deny
  9223372036854775807 system deny
process.max-data-size [ lowerable deny no-local-action ]
EXAMPLE 1 Displaying current resource control settings for a specific process (Continued)

```
18446744073709551615 privileged deny
18446744073709551615 system deny
process.max-file-size [ lowerable deny file-size ]
  9223372036854775807 privileged signal=XFSZ deny
  9223372036854775807 system deny
process.max-cpu-time [ lowerable no-deny cpu-time ]
  18446744073709551615 privileged signal=XCPU
  18446744073709551615 system deny [ infinite ]
task.max-cpu-time [ no-deny cpu-time ]
  18446744073709551615 system deny [ infinite ]
task.max-lwps
  2147483647 system deny
project.cpu-shares [ no-basic no-local-action ]
  10 privileged none
  65535 system deny
```

EXAMPLE 2 Displaying, replacing, and verifying the value of a specific control on an existing project

```
example# prctl -n project.cpu-shares -i project group.staff
111788: ksh
  project.cpu-shares [ no-basic no-local-action ]
    1 privileged none
    65535 system deny
# prctl -n project.cpu-shares -v 10 -r -i project group.staff
# prctl -n project.cpu-shares -i project group.staff
111788: ksh
  project.cpu-shares [ no-basic no-local-action ]
    10 privileged none
    65535 system deny
```

EXIT STATUS The following exit values are returned:

0  Success.
1  Fatal error encountered.
2  Invalid command line options were specified.

FILES /proc/pid/* process information and control files

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
</tbody>
</table>

SEE ALSO rctladm(1M), setrctl(2), rctlblk_get_local_action(3C), attributes(5)
preap(1)

NAME  
preap – force a defunct process to be reaped by its parent

SYNOPSIS  
preap [-F] pid...

DESCRIPTION  
A defunct (or zombie) process is one whose exit status has yet to be reaped by its
parent. The exit status is reaped via the wait(2), waitid(2), or waitpid(2) system
call. In the normal course of system operation, zombies may occur, but are typically
short-lived. This may happen if a parent exits without having reaped the exit status of
some or all of its children. In that case, those children are reparented to PID 1. See
init(1M), which periodically reaps such processes.

An irresponsible parent process may not exit for a very long time and thus leave
zombies on the system. Since the operating system destroys nearly all components of a
process before it becomes defunct, such defunct processes do not normally impact
system operation. However, they do consume a small amount of system memory.

preap forces the parent of the process specified by pid to waitid(2) for pid, if pid
represents a defunct process.

preap will attempt to prevent the administrator from unwisely reaping a child
process which might soon be reaped by the parent, if:

- The process is a child of init(1M).
- The parent process is stopped and might wait on the child when it is again allowed
to run.
- The process has been defunct for less than one minute.

OPTIONS  
The following option is supported:

- F      Forces the parent to reap the child, overriding safety checks.

OPERANDS  
The following operand is supported:

pid      Process ID list.

EXIT STATUS  
The following exit values are returned by preap, which prints the exit status of each
target process reaped:

0      Successfully operation.
non-zero Failure, such as no such process, permission denied, or invalid option.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu (32-bit)</td>
</tr>
<tr>
<td></td>
<td>SUNWesxu (64-bit)</td>
</tr>
</tbody>
</table>

SEE ALSO  
proc(1), init(1M), wait(2), waitid(2), waitpid(2), proc(4), attributes(5)
preap should be applied sparingly and only in situations in which the administrator or developer has confirmed that defunct processes will not be reaped by the parent process. Otherwise, applying preap may damage the parent process in unpredictable ways.
prex(1)

NAME
prex – control tracing and manipulate probe points in a process or the kernel

SYNOPSIS
prex [-o trace_file_name] [-l libraries] [-s kbytes_size] cmd [cmd-args...]
prex [-o trace_file_name] [-l libraries] [-s kbytes_size] -p pid
prex -k [-s kbytes_size]

DESCRIPTION
The prex command is the part of the Solaris tracing architecture that controls probes
in a process or the kernel. See tracing(3TNF) for an overview of this tracing
architecture, including example source code using it.

prex is the application used for external control of probes. It automatically preloads
the libtnfprobe library. prex locates all the probes in a target executable or the
kernel and provides an interface for the user to manipulate them. It allows a probe to
be turned on for tracing, debugging, or both. Tracing generates a TNF (Trace Normal
Form) trace file that can be converted to ASCII by tnfdump(1) and used for
performance analysis. Debugging generates a line to standard error whenever the
probe is hit at run time.

prex does not work on static executables. It only works on dynamic executables.

Invoking prex
There are three ways to invoke prex:

1. Use prex to start the target application cmd. In this case, the target application
   need not be built with a dependency on libtnfprobe. See TNF_PROBE(3TNF).
   prex sets the environment variable LD_PRELOAD to load libtnfprobe into the
target process. See ld(1). prex then uses the environment variable PATH to find
   the target application.

2. Attach prex to a running application. In this case, the running target application
   should have libtnfprobe already linked in. Alternatively, the user may
   manually set LD_PRELOAD to include libtnfprobe.so.1 prior to invoking the
   target.

3. Use prex with the -k option to set prex to kernel mode. prex can then be used
to control probes in the Solaris kernel. In kernel mode, additional commands are
defined, and some commands that are valid in other modes are invalid. See
Kernel Mode below.

Control File
Format and
Command
Language

In a future release of prex, the command language may be moved to a syntax that is
supported by an existing scripting language like ksh(1). In the meantime, the interface
to prex is uncommitted.

- Commands should be in ASCII.
- Each command is terminated with the NEWLINE character.
- A command can be continued onto the next line by ending the previous line with
  the backslash ("\") character.
- Tokens in a command must be separated by whitespace (one or more spaces or
tabs).
- The "#" character implies that the rest of the line is a comment.
% prex a.out
Attaches prex to your program and starts prex.

prex> enable $all
Enables all the probes.

prex> quit resume
Quits prex and resumes execution of program.

There are two different methods of communicating with prex:

- By specifications in a control file. During start-up, prex searches for a file named .prexrc in the directories specified below. prex does not stop at the first one it finds. This way a user can override any defaults that are set up. The search order is:
  
  $HOME/
  ./

- By typing commands at the prex prompt.

The command language for both methods is the same and is specified in USAGE. The commands that return output will not make sense in a control file. The output will go to standard output.

When using prex on a target process, the target will be in one of two states, running or stopped. This can be detected by the presence or absence of the prex> prompt. If the prompt is absent, it means that the target process is running. Typing Control-C will stop the target process and return the user to the prompt. There is no guarantee that Control-C will return to a prex prompt immediately. For example, if the target process is stopped on a job control stop (SIGSTOP), then Control-C in prex will wait until the target has been continued (SIGCONT). See Signals to Target Program below for more information on signals and the target process.

OPTIONS

The following options are supported:

- **-k**
  
  kernel mode: prex is used to control probes in the Solaris kernel. In kernel mode, additional commands are defined, and some commands valid in other modes are invalid. See Kernel Mode below.

- **-l libraries**
  
  The libraries mentioned are linked in to the target application using LD_PRELOAD (see ld(1)). This option cannot be used when attaching to a running process. The argument to the -l option should be a space-separated string enclosed in double quotes. Each token in the string is a library name. It follows the LD_PRELOAD rules on how libraries should be specified and where they will be found.
-o trace_file_name

File to be used for the trace output. trace_file_name is assumed to be relative to the current working directory of prex (that is, the directory that the user was in when prex was started).

If prex attaches to a process that is already tracing, the new trace_file_name (if provided) will not be used. If no trace_file_name is specified, the default is /$TMPDIR/trace-pid where pid is the process id of the target program. If TMPDIR is not set, /tmp is used.

-s kbytes_size

Maximum size of the output trace file in Kbytes. The default size of the trace kbytes_size is 4096 (2^12) bytes or 4 Mbytes for normal usage, and 384 or 384 kbytes in kernel mode. The minimum size that can be specified is 128 Kbytes. The trace file can be thought of as a least recently used circular buffer. Once the file has been filled, newer events will overwrite the older ones.

Grammar

Probes are specified by a list of space-separated selectors. Selectors are of the form:

attribute=value

(See TNF_PROBE(3TNF)). The “attribute=” is optional. If it is not specified, it defaults to “keys=”.

The attribute or value (generically called “spec”) can be any of the following:

IDENT Any sequence of letters, digits, _ , \ , ., % not beginning with a digit. IDENT implies an exact match.
QUOTED_STR Usually used to escape reserved words (any commands in the command language). QUOTED_STR implies an exact match and has to be enclosed in single quotes (’’).
REGEXP An ed(1) regular expression pattern match. REGEXP has to be enclosed in slashes (/ /), A / can be included in a REGEXP by escaping it with a backslash \. 

The following grammar explains the syntax.

selector_list ::= | /* empty */
selector_list selector
selector ::= spec=spec | /* whitespace around '=' opt */
spec ::= IDENT | QUOTED_STR | REGEXP

The terminals in the above grammar are:
This is a list of the remaining grammar that is needed to understand the syntax of the command language (defined in next subsection):

```
filename ::= QUOTED_STR  /* QUOTED_STR defined above */
spec_list ::= /* empty */ | spec_list spec /* spec defined above */
fcn_handle ::= &IDENT  /* IDENT defined above */
set_name ::= $IDENT  /* IDENT defined above */
```

1. Set Creation and Set Listing

```
create $set_name selector_list
list sets # list the defined sets
create can be used to define a set which contains probes that match the selector_list. The set $all is pre-defined as /.*/ and it matches all the probes.
```

2. Function Listing

```
list fcns  # list the available fcn_handle
```

The user can list the different functions that can be connected to probe points. Currently, only the debug function called &debug is available.

3. Commands to Connect and Disconnect Probe Functions

```
connect &fcn_handle $set_name
connect &fcn_handle selector_list
clear $set_name
clear selector_list
```

The `connect` command is used to connect probe functions (which must be prefixed by '&') to probes. The probes are specified either as a single set (with a '$'), or by explicitly listing the probe selectors in the command. The probe function has to be one that is listed by the `list fcns` command. This command does not enable the probes. The `clear` command is used to disconnect all connected probe functions from the specified probes.

4. Commands to Toggle the Tracing Mode

```
trace $set_name
trace selector_list
untrace $set_name
untrace selector_list
```

The `trace` and `untrace` commands are used to toggle the tracing action of a probe point (that is, whether a probe will emit a trace record or not if it is hit). This command does not enable the probes specified. Probes have tracing on by default. The most efficient way to turn off tracing is by using the `disable` command. `untrace` is useful if you want debug output but no tracing. If so, set the state of the probe to enabled, untraced, and the debug function connected.
5. Commands to Enable and Disable Probes

```bash
enable $set_name
enable selector_list
disable $set_name
disable selector_list
```

The `enable` and `disable` commands are used to control whether the probes perform the action that they have been set up for. To trace a probe, it has to be both enabled and traced (using the `trace` command). Probes are disabled by default. The `list history` command is used to list the probe control commands issued: `connect`, `clear`, `trace`, `untrace`, `enable`, and `disable`. These are the commands that are executed whenever a new shared object is brought in to the target program by `dlopen(3DL)`. See the subsection, `dlopen’ed Libraries`, below for more information. The following table shows the actions that result from specific combinations of tracing, enabling, and connecting:

<table>
<thead>
<tr>
<th>Enabled or Tracing State</th>
<th>Debug State</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disabled (On/Off)</td>
<td>(Connected/Cleared)</td>
<td>In</td>
</tr>
<tr>
<td>Enabled</td>
<td>On</td>
<td>Connected</td>
</tr>
<tr>
<td></td>
<td>Connected</td>
<td>Tracing and Debugging</td>
</tr>
<tr>
<td>Enabled</td>
<td>On</td>
<td>Cleared</td>
</tr>
<tr>
<td></td>
<td>Tracing only</td>
<td></td>
</tr>
<tr>
<td>Enabled</td>
<td>Off</td>
<td>Connected</td>
</tr>
<tr>
<td></td>
<td>Debugging only</td>
<td></td>
</tr>
<tr>
<td>Enabled</td>
<td>Off</td>
<td>Cleared</td>
</tr>
<tr>
<td></td>
<td>Nothing</td>
<td></td>
</tr>
<tr>
<td>Disabled</td>
<td>On</td>
<td>Connected</td>
</tr>
<tr>
<td></td>
<td>Nothing</td>
<td></td>
</tr>
<tr>
<td>Disabled</td>
<td>On</td>
<td>Cleared</td>
</tr>
<tr>
<td></td>
<td>Nothing</td>
<td></td>
</tr>
<tr>
<td>Disabled</td>
<td>Off</td>
<td>Connected</td>
</tr>
<tr>
<td></td>
<td>Nothing</td>
<td></td>
</tr>
<tr>
<td>Disabled</td>
<td>Off</td>
<td>Cleared</td>
</tr>
<tr>
<td></td>
<td>Nothing</td>
<td></td>
</tr>
</tbody>
</table>

6. List History

```bash
list history # lists probe control command history
```

The `list history` command displays a list of the probe control commands previously issued in the tracing session, for example, `connect`, `clear`, `trace`, `disable`. Commands in the history list are executed wherever a new shared object is brought into the target program by `dlopen(3DL)`.

7. Commands to List Probes, List Values, or List Trace File Name

```bash
list spec_list probes $set_name # list probes $all
list spec_list probes selector_list # list name probes file=test.c
list values spec_list # list values keys given in spec_list
list tracefile # list tracefile
```

The first two commands list the
selected attributes and values of the specified probes. They can be used to check
the state of a probe. The third command lists the various values associated with the
selected attributes. The fourth command lists the current trace file.

8. Help Command

help topic To get a list of the help topics that are available, invoke the help
command with no arguments. If a topic argument is specified, help is printed for
that topic.

9. Source a File

source filename The source command can be used to source a file of prex
commands. source can be nested (that is, a file can source another file). filename is
a quoted string.

10. Process Control

continue # resumes the target process
quit kill # quit prex, kill target
quit resume # quit prex, continue target
quit suspend # quit prex, leave target suspended
quit # quit prex (continue or kill target) The default quit will
continue the target process if prex attached to it. Instead, if prex had started the
target program, quit will kill the target process.

Probes in shared objects that are brought in by dlopen(3DL) are automatically set up
according to the command history of prex. When a shared object is removed by a
dclose(3DL), prex again needs to refresh its understanding of the probes in the
target program. This implies that there is more work to do for dlopen(3DL) and
dclose(3DL)—so they will take slightly longer. If a user is not interested in this
feature and doesn’t want to interfere with dlopen(3DL) and dclose(3DL), detach
prex from the target to inhibit this feature.

prex does not interfere with signals that are delivered directly to the target program.
However, prex receives all signals normally generated from the terminal, for
example, Control-C (SIGINT), and Control-Z (SIGSTOP), and does not forward them
to the target program. To signal the target program, use the kill(1) command from a
shell.

Process managing applications like dbx, truss(1), and prex cannot operate on the
same target program simultaneously. prex will not be able to attach to a target which
is being controlled by another application. A user can trace and debug a program
serially by the following method: first attach prex to target (or start target through
prex), set up the probes using the command language, and then type quit
suspend. The user can then attach dbx to the suspended process and debug it. A user
can also suspend the target by sending it a \texttt{SIGSTOP} signal, and then by typing \texttt{quit} \textit{resume} to \texttt{prex}. In this case, the user should also send a \texttt{SIGCONT} signal after invoking \texttt{dbx} on the stopped process (else \texttt{dbx} will be hung).

There are a few failure points that are possible when writing out events to a trace file, for example, system call failures. These failures result in a failure code being set in the target process. The target process continues normally, but no trace records are written. Whenever a user enters Control-C to \texttt{prex} to get to a \texttt{prex} prompt, \texttt{prex} will check the failure code in the target and inform the user if there was a tracing failure.

If the target program does a \texttt{fork(2)}, any probes that the child encounters will cause events to be logged to the same trace file. Events are annotated with a process id, so it will be possible to determine which process a particular event came from. In multi-threaded programs, there is a race condition with a thread doing a fork while the other threads are still running. For the trace file not to get corrupted, the user should either use \texttt{fork1(2)}, or make sure that all other threads are quiescent when doing a \texttt{fork(2)}.

If the target program itself (not any children it may \texttt{fork(2)}) does an \texttt{exec(2)}, \texttt{prex} detaches from the target and exits. The user can reconnect \texttt{prex} with \texttt{prex-p pid}.

A \texttt{vfork(2)} is generally followed quickly by an \texttt{exec(2)} in the child, and in the interim, the child borrows the parent’s process while the parent waits for the \texttt{exec(2)}. Any events logged by the child from the parent process will appear to have been logged by the parent.

Invoking \texttt{prex} with the \texttt{-k} flag causes \texttt{prex} to run in kernel mode. In kernel mode, \texttt{prex} controls probes in the Solaris kernel. See \texttt{tnf_kernel_probes(4)} for a list of available probes in the Solaris kernel. A few \texttt{prex} commands are unavailable in kernel mode; many other commands are valid in kernel mode only.

The \texttt{-l}, \texttt{-o}, and \texttt{-p} command-line options are not valid in kernel mode (that is, they may not be combined with the \texttt{-k} flag).

The rest of this section describes the differences in the \texttt{prex} command language when running \texttt{prex} in kernel mode.

1. \texttt{prex} will not stop the kernel

When \texttt{prex} attaches to a running user program, it stops the user program. Obviously, it cannot do this when attaching to the kernel. Instead, \texttt{prex} provides a “tracing master switch”: no probes will have any effect unless the tracing master switch is on. This allows the user to iteratively select probes to enable, then enable them all at once by turning on the master switch.

The command

\texttt{ktrace [ on | off ]} is used to inspect and set the value of the master switch.

Without an argument, \texttt{prex} reports the current state of the master switch. Since \texttt{prex} will not stop or kill the kernel, the
quit resume
and
quit kill commands are not valid in kernel mode.

2. No functions may be attached to probes in the kernel
   In particular, the debug function is unavailable in kernel mode.

3. Trace output is written to an in-core buffer
   In kernel mode, a trace output file is not generated directly, in order to allow
   probes to be placed in time-critical code. Instead, trace output is written to an
   in-core buffer, and copied out by a separate program, tnfextract(1).
   The in-core buffer is not automatically created. The following prex command
   controls buffer allocation and deallocation:

   buffer [ alloc [ size ] | dealloc ] Without an argument, the buffer
   command reports the size of the currently allocated buffer, if any. With an
   argument of alloc [size], prex allocates a buffer of the given size. size is in bytes,
   with an optional suffix of 'k' or 'm' specifying a multiplier of 1024 or 1048576,
   respectively. If no size is specified, the size specified on the command line with the
   -s option is used as a default. If the -s command line option was not used, the
   "default default" is 384 kilobytes. With an argument of dealloc, prex deallocates
   the trace buffer in the kernel. prex will reject attempts to turn the tracing master
   switch on when no buffer is allocated, and to deallocate the buffer when the tracing
   master switch is on. prex will refuse to allocate a buffer when one is already
   allocated; use buffer dealloc first. prex will not allocate a buffer larger than
   one-half of a machine's physical memory.

4. prex supports per-process probe enabling in the kernel
   In kernel mode, it is possible to select a set of processes for which probes are
   enabled. No trace output will be written when other processes traverse these probe
   points. This is called "process filter mode". By default, process filter mode is off,
   and all processes cause the generation of trace records when they hit an enabled
   probe.
   Some kernel events such as interrupts cannot be associated with a particular user
   process. By convention, these events are considered to be generated by process id
   0.

   prex provides commands to turn process filter mode on and off, to get the current
   status of the process filter mode switch, to add and delete processes (by process id)
   from the process filter set, and to list the current process filter set.

   The process filter set is maintained even when process filter mode is off, but has no
   effect unless process filter mode is on.
   When a process in the process filter set exits, its process id is automatically deleted
   from the process filter set.
   The command:
`pfilter` [ on | off | add pidlist | delete pidlist ] controls the process filter switch, and process filter set membership. With no arguments, `pfilter` prints the current process filter set and the state of the process filter mode switch:

- `on` or `off` set the state of the process filter mode switch.
- `add pidlist` add or delete processes from the process filter set. `pidlist` is a comma-separated list of one or more process ids.

**EXAMPLES**

See `tracing(3TNF)` for complete examples showing, among other things, the use of `prex` to do simple probe control.

When either the process or kernel is started, all probes are disabled.

**EXAMPLE 1** Set creation and set listing

- `create $out name=/out/` # $out = probes with "out" in value of "name" attribute
- `create $foo /page/ name=biodone` # $foo = union of probes with "page" in value of keys attribute and probes with "biodone" as value of "name" attribute
- `list sets` # list the defined sets
- `list fcns` # list the defined probe fcns

**EXAMPLE 2** Commands to trace and connect probe functions

- `trace foobar='on'` # exact match on foobar attribute
- `trace $all` # trace all probes (predefined set $all)
- `connect &debug $foo` # connect debug func to probes in $foo

**EXAMPLE 3** Commands to enable and disable probes

- `enable $all` # enable all probes
- `enable /vm/ name=alloc` # enable the specified probes
- `disable $foo` # disable probes in set $foo
- `list history` # list probe control commands issued

**EXAMPLE 4** Process control

- `continue` # resumes the target process
- `^C` # stop target; give control to prex
- `quit resume` # exit prex, leave process running and resume execution of program

**EXAMPLE 5** Kernel mode

- `buffer alloc 2m` # allocate a 2 Megabyte buffer
- `enable $all` # enable all probes
- `trace $all` # trace all probes
- `ktrace on` # turn tracing on
- `ktrace off` # turn tracing back off
EXAMPLE 5  Kernel mode  (Continued)

pfilter on  # turn process filter mode on
pfilter add 1379 # add pid 1379 to process filter
ktrace on  # turn tracing on
    # (only pid 1379 will be traced)

FILES
 .prexrc  local prex initialization file
-/prexrc  user's prex initialization file
/proc/nnnnn  process files

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWtnfc (32-bit)</td>
</tr>
<tr>
<td></td>
<td>SUNWtnfcx (64-bit)</td>
</tr>
</tbody>
</table>

SEE ALSO
ed(1), kill(1), ksh(1), ld(1), tnfdump(1), tnfxtract(1), truss(1), exec(2),
fork(2), fork1(2), vfork(2), TNF_DECLARE_RECORD(3TNF), TNF_PROBE(3TNF),
dlclose(3DL), dlopen(3DL), gethrtme(3C), libtnfctl(3TNF),
tnf_process_disable(3TNF), tracing(3TNF), tnf_kernel_probes(4),
attributes(5)

NOTES
Currently, the only probe function that is available is the &debug function. When this
function is executed, it prints out the arguments sent in to the probe as well as the
value associated with the sunw%debug attribute in the detail field (if any) to stderr.

For example, for the following probe point:

TNF_PROBE_2(input_values, "testapp main",
    "sunw%debug 'have read input values successfully'",
    tnf_long, int_input, x,
    tnf_string, string_input, input);

If x was 100 and input was the string "success", then the output of the debug probe
function would be:

probe input_values; sunw%debug "have read input values successfully";
int_input=100; string_input="success";

Some non-SPARC hardware lacks a true high-resolution timer, causing gethrtme() to return
the same value multiple times in succession. This can lead to problems in
how some tools interpret the trace file. This situation can be improved by interposing
a version of gethrtme(), which causes these successive values to be artificially
incremented by one nanosecond:
hrtime_t
gethrtime()
{
    static mutex_t lock;
    static hrtime_t (*real_gethrtime)(void) = NULL;
    static hrtime_t last_time = 0;
    hrtime_t this_time;
    if (real_gethrtime == NULL) {
        real_gethrtime =
            (hrtime_t (*)(void)) dlsym(RTLD_NEXT, "gethrtime");
    }
    this_time = real_gethrtime();
    mutex_lock(&lock);
    if (this_time <= last_time)
        this_time = ++last_time;
    else
        last_time = this_time;
    mutex_unlock(&lock);
    return (this_time);
}

Of course, this does not increase the resolution of the timer, so timestamps for
individual events are still relatively inaccurate. But this technique maintains ordering,
so that if event A causes event B, B never appears to happen before or at the same time
as A.

db x is available with the Sun Workshop Products.

BUGS
prex should issue a notification when a process id has been automatically deleted
from the filter set.

There is a known bug in prex which can result in this message:

Tracing shut down in target program due to an internal
error - Please restart prex and target

When prex runs as root, and the target process is not root, and the tracefile is placed
in a directory where it cannot be removed and re-created (a directory with the sticky
bit on, like /tmp), mm then the target process will not be able to open the tracefile
when it needs to. This results in tracing being disabled.

Changing any of the circumstances listed above should fix the problem. Either don’t
run prex as root, or run the target process as root, or specify the tracefile in a
directory other than /tmp.
print – shell built-in function to output characters to the screen or window

ksh  **print**  [-Rnprsu [n]]  [arg...]

ksh  The shell output mechanism. With no flags or with flag – or –, the arguments are printed on standard output as described by **echo**(1).

**OPTIONS**  The following options are supported:
- **-n**  suppresses new-line from being added to the output.
- **-R**  (raw mode) ignore the escape conventions of **echo**. The -R option will print all subsequent arguments and options other than -n.
- **-p**  causes the arguments to be written onto the pipe of the process spawned with | & instead of standard output.
- **-s**  causes the arguments to be written onto the history file instead of standard output.
- **-u [ n ]**  flag can be used to specify a one digit file descriptor unit number n on which the output will be placed. The default is 1.

**EXIT STATUS**  The following exit values are returned:
- 0  Successful operation.
- >0  Output file is not open for writing.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

**SEE ALSO**  **echo**(1), **ksh**(1), **attributes**(5)
NAME
printenv – display environment variables currently set

SYNOPSIS
/usr/ucb/printenv [variable]

DESCRIPTION
printenv prints out the values of the variables in the environment. If a variable is specified, only its value is printed.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

SEE ALSO
csh(1), echo(1), sh(1), stty(1), tset(1B), attributes(5), environ(5)

DIAGNOSTICS
If a variable is specified and it is not defined in the environment, printenv returns an exit status of 1.
NAME | printf – write formatted output  
SYNOPSIS | printf format [argument...]  
DESCRIPTION | The printf command writes formatted operands to the standard output. The argument operands are formatted under control of the format operand.  
OPERANDS | The following operands are supported:  
format | A string describing the format to use to write the remaining operands. The format operand is used as the format string described on the formats(5) manual page, with the following exceptions:  
- A SPACE character in the format string, in any context other than a flag of a conversion specification, is treated as an ordinary character that is copied to the output.  
- A character in the format string is treated as a character, not as a SPACE character.  
- In addition to the escape sequences described on the formats(5) manual page (\, \a, \b, \f, \n, \r, \t, \v), \ddd, where ddd is a one-, two- or three-digit octal number, is written as a byte with the numeric value specified by the octal number.  
- The program does not precede or follow output from the d or u conversion specifications with blank characters not specified by the format operand.  
- The program does not precede output from the o conversion specification with zeros not specified by the format operand.  
- An additional conversion character, b, is supported as follows. The argument is taken to be a string that may contain backslash-escape sequences. The following backslash-escape sequences are supported:  
  - the escape sequences listed on the formats(5) manual page (\, \a, \b, \f, \n, \r, \t, \v), which are converted to the characters they represent  
  - \ddd, where ddd is a zero-, one-, two- or three-digit octal number that is converted to a byte with the numeric value specified by the octal number  
  - \c, which is written and causes printf to ignore any remaining characters in the string operand containing it, any remaining string operands and any additional characters in the format operand.  
The interpretation of a backslash followed by any other sequence of characters is unspecified.
Bytes from the converted string are written until the end of the string or the number of bytes indicated by the precision specification is reached. If the precision is omitted, it is taken to be infinite, so all bytes up to the end of the converted string are written. For each specification that consumes an argument, the next argument operand is evaluated and converted to the appropriate type for the conversion as specified below. The format operand is reused as often as necessary to satisfy the argument operands. Any extra c or s conversion specifications are evaluated as if a null string argument were supplied; other extra conversion specifications are evaluated as if a zero argument were supplied. If the format operand contains no conversion specifications and argument operands are present, the results are unspecified. If a character sequence in the format operand begins with a % character, but does not form a valid conversion specification, the behavior is unspecified.

The strings to be written to standard output, under the control of format. The argument operands are treated as strings if the corresponding conversion character is b, c or s; otherwise, it is evaluated as a C constant, as described by the ISO C standard, with the following extensions:

- A leading plus or minus sign is allowed.
- If the leading character is a single- or double-quote, the value is the numeric value in the underlying codeset of the character following the single- or double-quote.

If an argument operand cannot be completely converted into an internal value appropriate to the corresponding conversion specification, a diagnostic message is written to standard error and the utility does not exit with a zero exit status, but continues processing any remaining operands and writes the value accumulated at the time the error was detected to standard output.

Note that this printf utility, like the printf(3C) function on which it is based, makes no special provision for dealing with multi-byte characters when using the %c conversion specification or when a precision is specified in a %b or %s conversion specification. Applications should be extremely cautious using either of these features when there are multi-byte characters in the character set.

Field widths and precisions cannot be specified as *.

For compatibility with previous versions of SunOS 5.x, the $ format specifier is supported for formats containing only %s specifiers.
The %b conversion specification is not part of the ISO C standard; it has been added here as a portable way to process backslash escapes expanded in string operands as provided by the `echo` utility. See also the USAGE section of the `echo(1)` manual page for ways to use `printf` as a replacement for all of the traditional versions of the `echo` utility.

If an argument cannot be parsed correctly for the corresponding conversion specification, the `printf` utility reports an error. Thus, overflow and extraneous characters at the end of an argument being used for a numeric conversion are to be reported as errors.

It is not considered an error if an argument operand is not completely used for a c or s conversion or if a string operand's first or second character is used to get the numeric value of a character.

**EXAMPLE 1** Printing a series of prompts

To alert the user and then print and read a series of prompts:

```bash
printf "\aPlease fill in the following: \nName: "
read name
printf "Phone number: "
read phone
```

**EXAMPLE 2** Printing a table of calculations

To read out a list of right and wrong answers from a file, calculate the percentage correctly, and print them out. The numbers are right-justified and separated by a single tab character. The percentage is written to one decimal place of accuracy:

```bash
while read right wrong ; do
    percent=$(echo "scale=1;($right*100)/($right+$wrong)" | bc)
    printf "%2d right\%2d wrong\%s%%\n"
        $right $wrong $percent
done < database_file
```

**EXAMPLE 3** Printing number strings

The command:

```bash
printf "%5d%4d\n" 1 21 321 4321 54321
```

produces:

```
   1 21
321 4321
54321  0
```

Note that the `format` operand is used three times to print all of the given strings and that a 0 was supplied by `printf` to satisfy the last `%d` conversion specification.
EXAMPLE 4 Tabulating conversion errors

The `printf` utility tells the user when conversion errors are detected while producing numeric output; thus, the following results would be expected on an implementation with 32-bit two's-complement integers when `%d` is specified as the `format` operand:

<table>
<thead>
<tr>
<th>Arguments</th>
<th>Standard</th>
<th>Diagnostic</th>
</tr>
</thead>
<tbody>
<tr>
<td>5a</td>
<td>5</td>
<td><code>printf: 5a not completely converted</code></td>
</tr>
<tr>
<td>9999999999999999999</td>
<td>2147483647</td>
<td><code>printf: 9999999999999999999: Results too large</code></td>
</tr>
<tr>
<td>-9999999999999999999</td>
<td>-2147483648</td>
<td><code>printf: -9999999999999999999: Results too large</code></td>
</tr>
<tr>
<td>ABC</td>
<td>0</td>
<td><code>printf: ABC expected numeric value</code></td>
</tr>
</tbody>
</table>

Note that the value shown on standard output is what would be expected as the return value from the function `strtol(3C)`. A similar correspondence exists between `%u` and `strtoul(3C)`, and `%e`, `%f` and `%g` and `strtod(3C)`.

EXAMPLE 5 Printing output for a specific locale

In a locale using the ISO/IEC 646:1991 standard as the underlying codeset, the command:

```c
printf "%d\n" 3 +3 -3 \"3 \"+3 \"-3"
```

produces:

<table>
<thead>
<tr>
<th></th>
<th>Numeric value of constant 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Numeric value of constant 3</td>
</tr>
<tr>
<td>-3</td>
<td>Numeric value of constant -3</td>
</tr>
<tr>
<td>51</td>
<td>Numeric value of the character ‘3’ in the ISO/IEC 646:1991 standard codeset</td>
</tr>
<tr>
<td>43</td>
<td>Numeric value of the character ‘+’ in the ISO/IEC 646:1991 standard codeset</td>
</tr>
<tr>
<td>45</td>
<td>Numeric value of the character ‘-‘ in the ISO/IEC 646:1991 standard codeset</td>
</tr>
</tbody>
</table>

Note that in a locale with multi-byte characters, the value of a character is intended to be the value of the equivalent of the `wchar_t` representation of the character.

If an argument operand cannot be completely converted into an internal value appropriate to the corresponding conversion specification, a diagnostic message is written to standard error and the utility does exit with a zero exit status, but continues processing any remaining operands and writes the value accumulated at the time the error was detected to standard output.
See environ(5) for descriptions of the following environment variables that affect the execution of printf: LC_COLLATE, LC_CTYPE, LC_MESSAGES, LC_TIME, TZ, and NLSPATH.

EXIT STATUS
The following exit values are returned:
0 Successful completion.
>0 An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWloc</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

SEE ALSO awk(1), bc(1), echo(1), printf(3C), strtod(3C), strtol(3C), strtoul(3C), attributes(5), environ(5), formats(5)
The `priocntl` command displays or sets scheduling parameters of the specified process(es). It can also be used to display the current configuration information for the system's process scheduler or execute a command with specified scheduling parameters.

Processes fall into distinct classes with a separate scheduling policy applied to each class. The process classes currently supported are the real-time class, time-sharing class, interactive class, fair-share class, and the fixed priority class. The characteristics of these classes and the class-specific options they accept are described below in the `USAGE` section under the headings `Real-Time Class`, `Time-Sharing Class`, `Interactive Class`, `Fair-Share Class`, and `Fixed-Priority Class`. With appropriate permissions, the `priocntl` command can change the class and other scheduling parameters associated with a running process.

In the default configuration, a runnable real-time process runs before any other process. Therefore, inappropriate use of real-time processes can have a dramatic negative impact on system performance.

If an `idlist` is present, it must appear last on the command line and the elements of the list must be separated by white space. If no `idlist` is present, an `idtype` argument of `pid`, `ppid`, `pgid`, `sid`, `taskid`, `class`, `uid`, `gid`, or `projid` specifies the process ID, parent process ID, process group ID, session ID, task ID, class, user ID, group ID, or project ID, respectively, of the `priocntl` command itself.

The command

```
priocntl -d [-i idtype] [idlist]
```

displays the class and class-specific scheduling parameters of the process(es) specified by `idtype` and `idlist`.

The command

```
priocntl -s [-c class] [class-specific options] \
       [-i idtype] [idlist]
```

sets the class and class-specific parameters of the specified processes to the values given on the command line. The `-c class` option specifies the class to be set. (The valid `class` arguments are `RT` for real-time, `TS` for time-sharing, `IA` for interactive, `FSS` for fair-share, or `FX` for fixed-priority.)
The class-specific parameters to be set are specified by the class-specific options as explained under the appropriate heading below. If the `-c class` option is omitted, `idtype` and `idlist` must specify a set of processes which are all in the same class, otherwise an error results. If no class-specific options are specified, the process's class-specific parameters are set to the default values for the class specified by `-c class` (or to the default parameter values for the process's current class if the `-c class` option is also omitted).

In order to change the scheduling parameters of a process using `priocntl` the real or effective user ID (respectively, groupID) of the user invoking `priocntl` must match the real or effective user ID (respectively, groupID) of the receiving process or the effective user ID of the user must be super-user. These are the minimum permission requirements enforced for all classes. An individual class may impose additional permissions requirements when setting processes to that class or when setting class-specific scheduling parameters.

When `idtype` and `idlist` specify a set of processes, `priocntl` acts on the processes in the set in an implementation-specific order. If `priocntl` encounters an error for one or more of the target processes, it may or may not continue through the set of processes, depending on the nature of the error.

If the error is related to permissions, `priocntl` prints an error message and then continues through the process set, resetting the parameters for all target processes for which the user has appropriate permissions. If `priocntl` encounters an error other than permissions, it does not continue through the process set but prints an error message and exits immediately.

A special `sys` scheduling class exists for the purpose of scheduling the execution of certain special system processes (such as the swapper process). It is not possible to change the class of any process to `sys`. In addition, any processes in the `sys` class that are included in the set of processes specified by `idtype` and `idlist` are disregarded by `priocntl`. For example, if `idtype` were `uid`, an `idlist` consisting of a zero would specify all processes with a UID of 0, except processes in the `sys` class and (if changing the parameters using the `-s` option) the `init` process.

The `init` process (process ID 1) is a special case. In order for the `priocntl` command to change the class or other scheduling parameters of the `init` process, `idtype` must be `pid` and `idlist` must be consist of only a 1. The `init` process may be assigned to any class configured on the system, but the time-sharing class is almost always the appropriate choice. (Other choices may be highly undesirable; see the `System Administration Guide: Basic Administration` for more information.)

The command

```bash
priocntl -e [-c class] [class-specific options] command \ 
    [argument...]```

executes the specified command with the class and scheduling parameters specified on the command line (arguments are the arguments to the command). If the `-c class` option is omitted the command is run in the user's current class.
The following options are supported:

- **-c class**  
  Specifies the class to be set. (The valid class arguments are `RT` for real-time, `TS` for time-sharing, `IA` for inter-active, `FSS` for fair-share, or `FX` for fixed-priority.) If the specified class is not already configured, it will be automatically configured.

- **-d**  
  Displays the scheduling parameters associated with a set of processes.

- **-e**  
  Executes a specified command with the class and scheduling parameters associated with a set of processes.

- **-i idtype**  
  This option, together with the `idlist` arguments (if any), specifies one or more processes to which the `priocntl` command is to apply. The interpretation of `idlist` depends on the value of `idtype`. The valid `idtype` arguments and corresponding interpretations of `idlist` are as follows:

  - **-ipid**  
    `idlist` is a list of process IDs. The `priocntl` command applies to the specified processes.

  - **-ippid**  
    `idlist` is a list of parent process IDs. The `priocntl` command applies to all processes whose parent process ID is in the list.

  - **-ipgid**  
    `idlist` is a list of process group IDs. The `priocntl` command applies to all processes in the specified process groups.

  - **-isid**  
    `idlist` is a list of session IDs. The `priocntl` command applies to all processes in the specified sessions.

  - **-itaskid**  
    `idlist` is a list of task IDs. The `priocntl` command applies to all processes in the specified tasks.

  - **-iclass**  
    `idlist` consists of a single class name (`RT` for real-time, `TS` for time-sharing, `IA` for inter-active, `FSS` for fair-share, or `FX` for fixed-priority). The `priocntl` command applies to all processes in the specified class.

  - **-iuid**  
    `idlist` is a list of user IDs. The `priocntl` command applies to all processes with an effective user ID equal to an ID from the list.

  - **-igid**  
    `idlist` is a list of group IDs. The `priocntl` command applies to all processes with an effective group ID equal to an ID from the list.
The priocntl command applies to all processes with an effective project ID equal to an ID from the list.

The priocntl command applies to all existing processes. No idlist should be specified (if one is specified, it is ignored). The permission restrictions described below still apply.

If the -i idtype option is omitted when using the -d or -s options the default idtype of pid is assumed.

Displays a list of the classes currently configured in the system along with class-specific information about each class. The format of the class-specific information displayed is described under USAGE.

Sets the scheduling parameters associated with a set of processes.

The valid class-specific options for setting real-time parameters are:

- Sets the real-time priority of the specified process(es) to rtpri.
- t t quantum [-r res] Sets the time quantum of the specified process(es) to tquantm. You may optionally specify a resolution as explained below.
- q t time quantum signal of the specified process(es) to tqsig.

The valid class-specific options for setting time-sharing parameters are:

- Sets the user priority limit of the specified process(es) to tsuprilim.
- Sets the user priority of the specified process(es) to tsupri.

The valid class-specific options for setting inter-active parameters are:

- Sets the user priority limit of the specified process(es) to iauprilim.
- Sets the user priority of the specified process(es) to iaupri.

The valid class-specific options for setting fair-share parameters are:

- Sets the user priority limit of the specified process(es) to fssuprilim.
- Sets the user priority of the specified process(es) to fssupri.

The valid class-specific options for setting fixed-priority parameters are:

- Sets the user priority limit of the specified process(es) to fxuprilim.
- Sets the user priority of the specified process(es) to fxupri.
The real-time class provides a fixed priority preemptive scheduling policy for those processes requiring fast and deterministic response and absolute user/application control of scheduling priorities. If the real-time class is configured in the system, it should have exclusive control of the highest range of scheduling priorities on the system. This ensures that a runnable real-time process is given CPU service before any process belonging to any other class.

The real-time class has a range of real-time priority (rtpr) values that may be assigned to processes within the class. Real-time priorities range from 0 to x, where the value of x is configurable and can be displayed for a specific installation that has already configured a real-time scheduler, by using the command

```
priocntl -l
```

The real-time scheduling policy is a fixed priority policy. The scheduling priority of a real-time process never changes except as the result of an explicit request by the user/application to change the rtpr value of the process.

For processes in the real-time class, the rtpr value is, for all practical purposes, equivalent to the scheduling priority of the process. The rtpr value completely determines the scheduling priority of a real-time process relative to other processes within its class. Numerically higher rtpr values represent higher priorities. Since the real-time class controls the highest range of scheduling priorities in the system, it is guaranteed that the runnable real-time process with the highest rtpr value is always selected to run before any other process in the system.

In addition to providing control over priority, priocntl provides for control over the length of the time quantum allotted to processes in the real-time class. The time quantum value specifies the maximum amount of time a process may run, assuming that it does not complete or enter a resource or event wait state (sleep). Notice that if another process becomes runnable at a higher priority, the currently running process may be preempted before receiving its full time quantum.

The command

```
priocntl -d [-i idtype] [idlist]
```

displays the real-time priority, time quantum (in millisecond resolution), and time quantum signal value for each real-time process in the set specified by idtype and idlist.

Any combination of the -p, -t, -r, and -q options may be used with priocntl -s or priocntl -e for the real-time class. If an option is omitted and the process is currently real-time, the associated parameter is unaffected. If an option is omitted when changing the class of a process to real-time from some other class, the associated parameter is set to a default value. The default value for rtpr is 0 and the default for time quantum is dependent on the value of rtpr and on the system configuration; see rt_dptbl(4).
When using the -t tqntm option, you may optionally specify a resolution using the -r res option. (If no resolution is specified, millisecond resolution is assumed.) If res is specified, it must be a positive integer between 1 and 1,000,000,000 inclusively and the resolution used is the reciprocal of res in seconds. For example, specifying -t 10 -r 100 would set the resolution to hundredths of a second and the resulting time quantum length would be 10/100 seconds (one tenth of a second). Although very fine (nanosecond) resolution may be specified, the time quantum length is rounded up by the system to the next integral multiple of the system clock’s resolution. Requests for time quantums of zero or quantums greater than the (typically very large) implementation-specific maximum quantum result in an error.

The real-time time quantum signal can be used to notify runaway real-time processes about the consumption of their time quantum. Those processes, which are monitored by the real-time time quantum signal, receive the configured signal in the event of time quantum expiration. The default value (0) of the time quantum signal tqsig will denote no signal delivery. A positive value will denote the delivery of the signal specified by the value. Like kill(1) and other commands operating on signals, the -q tqsig option is also able to handle symbolically named signals, like XCPU or KILL.

In order to change the class of a process to real-time (from any other class), the user invoking priocntl must have super-user privilege. In order to change the rtpri value or time quantum of a real-time process, the user invoking priocntl must either be super-user, or must currently be in the real-time class (shell running as a real-time process) with a real or effective user ID matching the real or effective user ID of the target process.

The real-time priority, time quantum, and time quantum signal are inherited across the fork(2) and exec(2) system calls. When using the time quantum signal with a user defined signal handler across the exec(2) system call, the new image must install an appropriate user defined signal handler before the time quantum expires. Otherwise, unpredictable behavior would result.

The time-sharing scheduling policy provides for a fair and effective allocation of the CPU resource among processes with varying CPU consumption characteristics. The objectives of the time-sharing policy are to provide good response time to interactive processes and good throughput to CPU-bound jobs, while providing a degree of user/application control over scheduling.

The time-sharing class has a range of time-sharing user priority (tsupri) values that may be assigned to processes within the class. User priorities range from −x to +x, where the value of x is configurable. The range for a specific installation can be displayed by using the command

```
priocntl -l
```

The purpose of the user priority is to provide some degree of user/application control over the scheduling of processes in the time-sharing class. Raising or lowering the tsupri value of a process in the time-sharing class raises or lowers the scheduling...
priority of the process. It is not guaranteed, however, that a time-sharing process with a higher \textit{tsupri} value will run before one with a lower \textit{tsupri} value. This is because the \textit{tsupri} value is just one factor used to determine the scheduling priority of a time-sharing process. The system may dynamically adjust the internal scheduling priority of a time-sharing process based on other factors such as recent CPU usage.

In addition to the system-wide limits on user priority (displayed with \texttt{priocntl -l}), there is a per process user priority limit (\textit{tsuprilim}), which specifies the maximum \textit{tsupri} value that may be set for a given process.

The command

\texttt{priocntl -d [-i idtype] [idlist]}

displays the user priority and user priority limit for each time-sharing process in the set specified by \texttt{idtype} and \texttt{idlist}.

Any time-sharing process may lower its own \textit{tsuprilim} (or that of another process with the same user ID). Only a time-sharing process with super-user privilege may raise a \textit{tsuprilim}. When changing the class of a process to time-sharing from some other class, super-user privilege is required in order to set the initial \textit{tsuprilim} to a value greater than zero.

Any time-sharing process may set its own \textit{tsupri} (or that of another process with the same user ID) to any value less than or equal to the process’s \textit{tsuprilim}. Attempts to set the \textit{tsupri} above the \textit{tsuprilim} (and/or set the \textit{tsuprilim} below the \textit{tsupri}) result in the \textit{tsupri} being set equal to the \textit{tsuprilim}.

Any combination of the \texttt{-m} and \texttt{-p} options may be used with \texttt{priocntl -s} or \texttt{priocntl -e} for the time-sharing class. If an option is omitted and the process is currently time-sharing, the associated parameter is normally unaffected. The exception is when the \texttt{-p} option is omitted and \texttt{-m} is used to set a \textit{tsuprilim} below the current \textit{tsupri}. In this case, the \textit{tsupri} is set equal to the \textit{tsuprilim} which is being set. If an option is omitted when changing the class of a process to time-sharing from some other class, the associated parameter is set to a default value. The default value for \textit{tsuprilim} is 0 and the default for \textit{tsupri} is to set it equal to the \textit{tsuprilim} value which is being set.

The time-sharing user priority and user priority limit are inherited across the \texttt{fork(2)} and \texttt{exec(2)} system calls.

**Inter-Active Class**

The inter-active scheduling policy provides for a fair and effective allocation of the CPU resource among processes with varying CPU consumption characteristics while providing good responsiveness for user interaction. The objectives of the inter-active policy are to provide good response time to interactive processes and good throughput to CPU-bound jobs. The priorities of processes in the inter-active class can be changed in the same manner as those in the time-sharing class, though the modified priorities will continue to be adjusted to provide good responsiveness for user interaction.
The inter-active user priority limit, iaupri, is equivalent to tsupri. The inter-active per process user priority, iauprilim, is equivalent to tsuprilim.

Inter-active class processes that have the iamode ("interactive mode") bit set are given a priority boost value of 10, which is factored into the user mode priority of the process when that calculation is made, that is, every time a process’s priority is adjusted. This feature is used by the X windowing system, which sets this bit for those processes that run inside of the current active window to give them a higher priority.

Fair-Share Class

The fair-share scheduling policy provides a fair allocation of system CPU resources among projects, independent of the number of processes they own. Projects are given "shares" to control their entitlement to CPU resources. Resource usage is remembered over time, so that entitlement is reduced for heavy usage, and increased for light usage, with respect to other projects. CPU time is scheduled among processes according to their owner’s entitlements, independent of the number of processes each project owns.

The FSS scheduling class supports the notion of per-process user priority and user priority limit for compatibility with the time-share scheduler. The fair share scheduler attempts to provide an evenly graded effect across the whole range of user priorities. Processes with positive fssupri values receive time slices less frequently than normal, while negative nice processes receive time slices more frequently than normal. Notice that user priorities do not interfere with shares. That is, changing a fssupri value of a process is not going to affect its project’s overall CPU usage which only relates to the amount of shares it is allocated compared to other projects.

The priorities of processes in the fair-share class can be changed in the same manner as those in the time-share class.

Fixed-Priority Class

The fixed-priority class provides a fixed priority preemptive scheduling policy for those processes requiring that the scheduling priorities do not get dynamically adjusted by the system and that the user/application have control of the scheduling priorities.

The fixed-priority class shares the same range of scheduling priorities with the time-sharing class, by default. The fixed-priority class has a range of fixed-priority user priority (fxupri) values that may be assigned to processes within the class. User priorities range from 0 to x, where the value of x is configurable. The range for a specific installation can be displayed by using the command

```bash
priocntl -l
```

The purpose of the user priority is to provide user/application control over the scheduling of processes in the fixed-priority class. For processes in the fixed-priority class, the fxupri value is, for all practical purposes, equivalent to the scheduling priority of the process. The fxupri value completely determines the scheduling priority of a fixed-priority process relative to other processes within its class. Numerically higher fxupri values represent higher priorities.
In addition to the system-wide limits on user priority (displayed with `priocntl -l`), there is a per process user priority limit (`fxuprilim`), which specifies the maximum `fxupri` value that may be set for a given process.

Any fixed-priority process may lower its own `fxuprilim` (or that of another process with the same user ID). Only a process with super-user privilege may raise a `fxuprilim`. When changing the class of a process to fixed-priority from some other class, super-user privilege is required in order to set the initial `fxuprilim` to a value greater than zero.

Any fixed-priority process may set its own `fxupri` (or that of another process with the same user ID) to any value less than or equal to the process’ `fxuprilim`. Attempts to set the `fxupri` above the `fxuprilim` (and/or set the `fxuprilim` below the `fxupri`) result in the `fxupri` being set equal to the `fxuprilim`.

In addition to providing control over priority, `priocntl` provides for control over the length of the time quantum allotted to processes in the fixed-priority class. The time quantum value specifies the maximum amount of time a process may run, before surrendering the CPU, assuming that it does not complete or enter a resource or event wait state (sleep). Notice that if another process becomes runnable at a higher priority, the currently running process may be preempted before receiving its full time quantum.

Any combination of the `-m`, `-p`, and `-t` options may be used with `priocntl -s` or `priocntl -e` for the fixed-priority class. If an option is omitted and the process is currently fixed-priority, the associated parameter is normally unaffected. The exception is when the `-p` option is omitted and the `-m` option is used to set a `fxuprilim` below the current `fxupri`. In this case, the `fxupri` is set equal to the `fxuprilim` which is being set. If an option is omitted when changing the class of a process to fixed-priority from some other class, the associated parameter is set to a default value. The default value for `fxuprilim` is 0. The default for `fxupri` is to set it equal to the `fxuprilim` value which is being set. The default for time quantum is dependent on the `fxupri` and on the system configuration. See `fx_dptbl(4)`.

The time quantum of processes in the fixed-priority class can be changed in the same manner as those in the real-time class.

The fixed-priority user priority, user priority limit, and time quantum are inherited across the `fork(2)` and `exec(2)` system calls.

**EXAMPLES**

**Real-Time Class examples follow:**

**EXAMPLE 1** Setting the class of any non-real-time processes

This example sets the class of any non-real-time processes selected by `idtype` and `idlist` to real-time and sets their real-time priority to the default value of 0. The real-time priorities of any processes currently in the real-time class are unaffected. The time quantums of all of the specified processes are set to 1/10 seconds.
EXAMPLE 1 Setting the class of any non-real-time processes (Continued)

exampel@ priocntl -s -c RT -t 1 -r 10 -i idtype idlist

EXAMPLE 2 Executing a command in real-time

This example executes `command` in the real-time class with a real-time priority of 15 and a time quantum of 20 milliseconds:

`example@ priocntl -e -c RT -p 15 -t 20 command`

EXAMPLE 3 Executing a command in real-time with a specified quantum signal

This example executes `command` in the real-time class with a real-time priority of 11, a time quantum of 250 milliseconds, and where the specified real-time quantum signal is SIGXCPU:

`example@ priocntl -e -c RT -p 11 -t 250 -q XCPU command`

Time-Sharing Class examples follow:

EXAMPLE 4 Setting the class of non-time-sharing processes

This example sets the class of any non-time-sharing processes selected by `idtype` and `idlist` to time-sharing and sets both their user priority limit and user priority to 0. Processes already in the time-sharing class are unaffected.

`example@ priocntl -s -c TS -i idtype idlist`

EXAMPLE 5 Executing a command in the time-sharing class

This example executes `command` with the arguments `arguments` in the time-sharing class with a user priority limit of 0 and a user priority of −15:

`example@ priocntl -e -c TS -m 0 -p -15 command [arguments]`

EXAMPLE 6 Executing a command in fixed-priority class

This example executes a command in the fixed-priority class with a user priority limit of 20 and user priority of 10 and time quantum of 250 milliseconds:

`example@ priocntl -e -c FX -m 20 -p 10 -t 250 command`

EXIT STATUS

The following exit values are returned:

For options `-d`, `-l`, and `-s`:

0 Successful operation.
priocntl(1)

1 Error condition.

For option -e:

Return of the Exit Status of the executed command denotes successful operation. Otherwise,

1 Command could not be executed at the specified priority.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

SEE ALSO

kill(1), nice(1), ps(1), exec(2), fork(2), priocntl(2), fx_dptbl(4), rt_dptbl(4), attributes(5), FSS(7)

System Administration Guide: Basic Administration

DIAGNOSTICS

priocntl prints the following error messages:

Process(es) not found
None of the specified processes exists.

Specified processes from different classes
The -s option is being used to set parameters, the -c class option is not present, and processes from more than one class are specified.

Invalid option or argument
An unrecognized or invalid option or option argument is used.
proc(1)

NAME
proc, pflags, pcrcd, pldd, psig, pstack, pfiles, pwdx, pstop, prun, pwait, ptree, ptime – proc tools

SYNOPSIS
/usr/bin/pflags [-r] [pid | core...]
/usr/bin/pcred [pid | core...]
/usr/bin/pldd [-F] [pid | core...]
/usr/bin/psig [-n] pid...
/usr/bin/pstack [-F] [pid | core...]
/usr/bin/pfiles [-F] pid...
/usr/bin/pwdx [-F] pid...
/usr/bin/pstop pid...
/usr/bin/prun pid...
/usr/bin/pwait [-v] pid...
/usr/bin/ptree [-a] [pid | user...]
/usr/bin/ptime command [arg...]

DESCRIPTION
The proc tools are utilities that exercise features of /proc (see proc(4)). Most of them take a list of process-ids (pid). The tools that do take process-ids also accept /proc/nnn as a process-id, so the shell expansion /proc/* can be used to specify all processes in the system.

Some of the proc tools can also be applied to core files (see core(4)). The tools that apply to core files accept a list of either process IDs or names of core files or both.

pflags
Print the /proc tracing flags, the pending and held signals, and other /proc status information for each lwp in each process.

pcrcd
Print the credentials (effective, real, saved UIDs and GIDs) of each process.

pldd
List the dynamic libraries linked into each process, including shared objects explicitly attached using dlopen(3DL). See also ldd(1).

psig
List the signal actions and handlers of each process. See signal(3HEAD).

pstack
Print a hex+symbolic stack trace for each lwp in each process.

pfiles
Report fstat(2) and fcntl(2) information for all open files in each process.

pwdx
Print the current working directory of each process.

pstop
Stop each process (PR_REQUESTED stop).

prun
Set each process running (inverse of pstop).
proc(1)

**pw**\textit{ait}  
Wait for all of the specified processes to terminate.

**ptree**  
Print the process trees containing the specified pids or users, with child processes indented from their respective parent processes. 
An argument of all digits is taken to be a process-id, otherwise it is assumed to be a user login name. Default is all processes.

**ptime**  
Time the command, like time(1), but using microstate accounting for reproducible precision. Unlike time(1), children of the command are not timed.

**OPTIONS**  
The following options are supported:

- **-a**  
  (ptree only) All. Includes children of process 0.

- **-F**  
  Force. Grabs the target process even if another process has control.

- **-n**  
  (psig only) Displays signal handler addresses rather than names.

- **-r**  
  (pflags only) If the process is stopped, displays its machine registers.

- **-v**  
  (pwait only) Verbose. Reports terminations to standard output.

**USAGE**  
These proc tools stop their target processes while inspecting them and reporting the results: pfiles, pldd, and pstack. A process can do nothing while it is stopped. Thus, for example, if the X server is inspected by one of these proc tools running in a window under the X server’s control, the whole window system can become deadlocked because the proc tool would be attempting to print its results to a window that cannot be refreshed. Logging in from another system using rlogin(1) and killing the offending proc tool would clear up the deadlock in this case.

Caution should be exercised when using the -F flag. Imposing two controlling processes on one victim process can lead to chaos. Safety is assured only if the primary controlling process, typically a debugger, has stopped the victim process and the primary controlling process is doing nothing at the moment of application of the proc tool in question.

Some of the proc tools can also be applied to core files, as shown by the synopsis above. A core file is a snapshot of a process’s state and is produced by the kernel prior to terminating a process with a signal or by the gcore(1) utility. Some of the proc tools may need to derive the name of the executable corresponding to the process which dumped core or the names of shared libraries associated with the process. These files are needed, for example, to provide symbol table information for pstack(1). If the proc tool in question is unable to locate the needed executable or shared library, some symbol information will be unavailable for display. Similarly, if a core file from one operating system release is examined on a different operating system release, the run-time link-editor debugging interface (librtld_db) may not be able to initialize. In this case, symbol information for shared libraries will not be available.

**EXIT STATUS**  
The following exit values are returned:
Successful operation.
non-zero An error has occurred.

FILES
/proc/*          process files
/usr/proc/lib/*  proc tools supporting files

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu (32-bit)</td>
</tr>
<tr>
<td></td>
<td>SUNWesxu (64-bit)</td>
</tr>
</tbody>
</table>

SEE ALSO
gcore(1), ldd(1), pargs(1), pgrep(1), pkill(1), pmap(1), preap(1),
p(1), pwd(1), rlogin(1), time(1), truss(1), wait(1), fcntl(2), fstat(2),
dlopen(3DL), signal(3HEAD), core(4), proc(4), attributes(5)
The `prof` command interprets a profile file produced by the `monitor` function. The symbol table in the object file `prog` (a.out by default) is read and correlated with a profile file (mon.out by default). For each external text symbol the percentage of time spent executing between the address of that symbol and the address of the next is printed, together with the number of times that function was called and the average number of milliseconds per call.

**OPTIONS**

The mutually exclusive options `-a`, `-c`, `-n`, and `-t` determine the type of sorting of the output lines:

- `-a` Sort by increasing symbol address.
- `-c` Sort by decreasing number of calls.
- `-n` Sort lexically by symbol name.
- `-t` Sort by decreasing percentage of total time (default).

The mutually exclusive options `-o` and `-x` specify the printing of the address of each symbol monitored:

- `-o` Print each symbol address (in octal) along with the symbol name.
- `-x` Print each symbol address (in hexadecimal) along with the symbol name.

The mutually exclusive options `-g` and `-l` control the type of symbols to be reported. The `-l` option must be used with care; it applies the time spent in a static function to the preceding (in memory) global function, instead of giving the static function a separate entry in the report. If all static functions are properly located, this feature can be very useful. If not, the resulting report may be misleading.

Assume that A and B are global functions and only A calls static function S. If S is located immediately after A in the source code (that is, if S is properly located), then, with the `-l` option, the amount of time spent in A can easily be determined, including the time spent in S. If, however, both A and B call S, then, if the `-l` option is used, the report will be misleading; the time spent during B’s call to S will be attributed to A, making it appear as if more time had been spent in A than really had. In this case, function S cannot be properly located.

- `-g` List the time spent in static (non-global) functions separately. The `-g` option function is the opposite of the `-l` function.
- `-l` Suppress printing statically declared functions. If this option is given, time spent executing in a static function is allocated to the closest global function loaded before the static function in the executable. This option is the default. It is the opposite of the `-g` function and should be used with care.

The following options may be used in any combination:
Demangle C++ symbol names before printing them out.

Suppress the heading normally printed on the report. This is useful if the report is to be processed further.

Use file mdata instead of mon.out as the input profile file.

Print a summary of several of the monitoring parameters and statistics on the standard error output.

Print prof version information on the standard error output.

Include all symbols in the profile range, even if associated with zero number of calls and zero time.

A program creates a profile file if it has been link edited with the -p option of cc(1B). This option to the cc(1B) command arranges for calls to monitor at the beginning and end of execution. It is the call to monitor at the end of execution that causes the system to write a profile file. The number of calls to a function is tallied if the -p option was used when the file containing the function was compiled.

A single function may be split into subfunctions for profiling by means of the MARK macro. See prof(5).

The name of the file created by a profiled program is controlled by the environment variable PROFDIR. If PROFDIR is not set, mon.out is produced in the directory current when the program terminates. If PROFDIR=string, string/pid.progname is produced, where progname consists of argv[0] with any path prefix removed, and pid is the process ID of the program. If PROFDIR is set, but null, no profiling output is produced.

The times reported in successive identical runs may show variances because of varying cache-hit ratios that result from sharing the cache with other processes. Even if a program seems to be the only one using the machine, hidden background or asynchronous processes may blur the data. In rare cases, the clock ticks initiating recording of the program counter may ‘beat’ with loops in a program, grossly distorting measurements. Call counts are always recorded precisely, however.

```
ENVIRONMENT VARIABLES
PROFDIR The name of the file created by a profiled program is controlled by the
environment variable PROFDIR. If PROFDIR is not set, mon.out is produced in
the directory current when the program terminates. If PROFDIR=string, string/pid.progname is produced,
where progname consists of argv[0] with any path prefix removed, and pid is the process ID of the program. If PROFDIR is
set, but null, no profiling output is produced.
FILES
mon.out default profile file
a.out default namelist (object) file
ATTRIBUTES See attributes(5) for descriptions of the following attributes:

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWbtool</td>
</tr>
</tbody>
</table>
SEE ALSO cc(1B), gprof(1), exit(2), pcsample(2), profil(2), malloc(3C), malloc(3MALLOC), monitor(3C), attributes(5), prof(5)
NOTES The times reported in successive identical runs may show variances because of varying cache-hit ratios that result from sharing the cache with other processes. Even if a program seems to be the only one using the machine, hidden background or asynchronous processes may blur the data. In rare cases, the clock ticks initiating recording of the program counter may ‘beat’ with loops in a program, grossly distorting measurements. Call counts are always recorded precisely, however.
```
Only programs that call `exit` or return from `main` are guaranteed to produce a profile file, unless a final call to `monitor` is explicitly coded.

The times for static functions are attributed to the preceding external text symbol if the `-g` option is not used. However, the call counts for the preceding function are still correct; that is, the static function call counts are not added to the call counts of the external function.

If more than one of the options `-t`, `-c`, `-a`, and `-n` is specified, the last option specified is used and the user is warned.

`LD_LIBRARY_PATH` must not contain `/usr/lib` as a component when compiling a program for profiling. If `LD_LIBRARY_PATH` contains `/usr/lib`, the program will not be linked correctly with the profiling versions of the system libraries in `/usr/lib/libp`. See `gprof(1)`.

Functions such as `mcount()`, `_mcount()`, `moncontrol()`, `_moncontrol()`, `monitor()`, and `_monitor()` may appear in the `prof` report. These functions are part of the profiling implementation and thus account for some amount of the runtime overhead. Since these functions are not present in an unprofiled application, time accumulated and call counts for these functions may be ignored when evaluating the performance of an application.

### 64-bit profiling

64-bit profiling may be used freely with dynamically linked executables, and profiling information is collected for the shared objects if the objects are compiled for profiling. Care must be applied to interpret the profile output, since it is possible for symbols from different shared objects to have the same name. If duplicate names are seen in the profile output, it is better to use the `-s` (summary) option, which prefixes a module id before each symbol that is duplicated. The symbols can then be mapped to appropriate modules by looking at the modules information in the summary.

If the `-a` option is used with a dynamically linked executable, the sorting occurs on a per-shared-object basis. Since there is a high likelihood of symbols from different shared objects to have the same value, this results in an output that is more understandable. A blank line separates the symbols from different shared objects, if the `-s` option is given.

### 32-bit profiling

32-bit profiling may be used with dynamically linked executables, but care must be applied. In 32-bit profiling, shared objects cannot be profiled with `prof`. Thus, when a profiled, dynamically linked program is executed, only the "main" portion of the image is sampled. This means that all time spent outside of the "main" object, that is, time spent in a shared object, will not be included in the profile summary; the total time reported for the program may be less than the total time used by the program.

Because the time spent in a shared object cannot be accounted for, the use of shared objects should be minimized whenever a program is profiled with `prof`. If desired, the program should be linked to the profiled version of a library (or to the standard
archive version if no profiling version is available), instead of the shared object to get
profile information on the functions of a library. Versions of profiled libraries may be
supplied with the system in the /usr/lib/libp directory. Refer to compiler driver
documentation on profiling.

Consider an extreme case. A profiled program dynamically linked with the shared C
library spends 100 units of time in some libc routine, say, malloc(). Suppose
malloc() is called only from routine B and B consumes only 1 unit of time. Suppose
further that routine A consumes 10 units of time, more than any other routine in the
"main" (profiled) portion of the image. In this case, prof will conclude that most of
the time is being spent in A and almost no time is being spent in B. From this it will be
almost impossible to tell that the greatest improvement can be made by looking at
routine B and not routine A. The value of the profiler in this case is severely degraded;
the solution is to use archives as much as possible for profiling.
The `profiles` command prints on standard output the names of the execution profiles that have been assigned to you or to the optionally-specified user or role name. Profiles are a bundling mechanism used to enumerate the commands and authorizations needed to perform a specific function. Along with each listed executable are the process attributes, such as the effective user and group IDs, with which the process runs when started by a privileged command interpreter. The profile shells are `pfcsh`, `pfksh`, and `pfexec`. See the `pfexec(1)` man page. Profiles can contain other profiles defined in `prof_attr(4)`.

Multiple profiles can be combined to construct the appropriate access control. When profiles are assigned, the authorizations are added to the existing set. If the same command appears in multiple profiles, the first occurrence, as determined by the ordering of the profiles, is used for process-attribute settings. For convenience, a wildcard can be specified to match all commands.

When profiles are interpreted, the profile list is loaded from `user_attr(4)`. If any default profile is defined in `/etc/security/policy.conf` (see `policy.conf(4)`), the list of default profiles will be added to the list loaded from `user_attr(4)`. Matching entries in `prof_attr(4)` provide the authorizations list, and matching entries in `exec_attr(4)` provide the commands list.

**OPTIONS**

- `-l` Lists the commands in each profile followed by the special process attributes such as user and group IDs.

**EXAMPLES**

**EXAMPLE 1** Sample output

The output of the `profiles` command has the following form:

```
example% profiles tester01 tester02
 tester01 : Audit Management, All Commands
tester02 : Device Management, All Commands
example%
```

**EXAMPLE 2** Using the list option

```
example% profiles -l tester01 tester02
 tester01 :
 Audit Management:
 /usr/sbin/audit euid=root
 /usr/sbin/auditconfig euid=root egid=sys
 All Commands:
 *

 tester02 :
 Device Management:
 /usr/bin/allocate: euid=root
 /usr/bin/deallocate: euid=root
 All Commands
 *

 example%
```
EXAMPLE 2 Using the list option  

(Continued)

EXIT STATUS The following exit values are returned:

0 Successful completion.
1 An error occurred.

FILES
/etc/security/exec_attr
/etc/security/prof_attr
/etc/user_attr
/etc/security/policy.conf

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO auths(1), pfexec(1), roles(1), getprofattr(3SECDB), exec_attr(4), policy.conf(4), prof_attr(4), user_attr(4), attributes(5)
NAME  projects – print project membership of user

SYNOPSIS  projects [-dv] [ user]

DESCRIPTION  The projects command prints on standard output the projects to which the
invoking user or an optionally specified user belongs. Each user belongs to some set of
projects specified in the project(4) file and possibly in the associated NIS maps and
LDAP databases for project information.

OPTIONS  The following options are supported:
-d    Prints only default project.
-v    Prints project descriptions along with project names.

OPERANDS  The following operand is supported:
user    Displays project memberships for the specified user.

EXAMPLES  EXAMPLE 1 Displaying membership for a specified user
example$ projects paul
default beatles wings
example$ projects ringo
default beatles
example$ projects -d paul
beatles

EXIT STATUS  The following exit values are returned:
0    Successful completion.
1    A fatal error occurred during execution.
2    Invalid command line options were specified.

FILES  /etc/project    Local database containing valid project definitions for this
machine.

ATTRIBUTES  See attributes(5) for a description of the following attributes:

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO  getdefaultproj(3PROJECT), getprojent(3PROJECT), project(4),
attributes(5)
NAME

ps – report process status

SYNOPSIS

ps [-AcdeljLPy] [-g grplist [-n namelist] [-o format] ... [-p proclist]
     [-s sidlist] [-t term] [-u uidlist] [-U uidlist] [-G gidlist]

DESCRIPTION

The ps command prints information about active processes. Without options, ps
prints information about processes that have the same effective user ID and the same
controlling terminal as the invoker. The output contains only the process ID, terminal
identifier, cumulative execution time, and the command name. Otherwise, the
information that is displayed is controlled by the options.

Some options accept lists as arguments. Items in a list can be either separated by
commas or else enclosed in quotes and separated by commas or spaces. Values for
proclist and grplist must be numeric.

OPTIONS

The following options are supported:

-a

Lists information about all processes most frequently requested:
all those except session leaders and processes not associated with a
terminal.

-A

Lists information for all processes. Identical to -e, below.

-c

Prints information in a format that reflects scheduler properties as
described in priocntl(1). The -c option affects the output of the
-f and -l options, as described below.

-d

Lists information about all processes except session leaders.

-e

Lists information about every process now running.

-f

Generates a full listing. (See below for significance of columns in a
full listing.)

-g grplist

Lists only process data whose group leader’s ID number(s)
appears in grplist. (A group leader is a process whose process ID
number is identical to its process group ID number.)

-G gidlist

Lists information for processes whose real group ID numbers are
given in gidlist. The gidlist must be a single argument in the form
of a blank- or comma-separated list.

-j

Prints session ID and process group ID.

-l

Generates a long listing. (See below.)

-L

Prints information about each light weight process (lwp) in each
selected process. (See below.)

-n namelist

Specifies the name of an alternative system namelist file in place of
the default. This option is accepted for compatibility, but is
ignored.

-o format

Prints information according to the format specification given in
format. This is fully described in DISPLAY FORMATS. Multiple -o
options can be specified; the format specification will be interpreted as the space-character-separated concatenation of all the format option-arguments.

-\texttt{ps proclist} Lists only process data whose process ID numbers are given in \texttt{proclist}.

-\texttt{P} Prints the number of the processor to which the process or lwp is bound, if any, under an additional column header, \texttt{PSR}.

-\texttt{s sidlist} Lists information on all session leaders whose IDs appear in \texttt{sidlist}.

-\texttt{t term} Lists only process data associated with \texttt{term}. Terminal identifiers are specified as a device file name, and an identifier. For example, \texttt{term/a}, or \texttt{pts/0}.

-\texttt{u uidlist} Lists only process data whose effective user ID number or login name is given in \texttt{uidlist}. In the listing, the numerical user ID will be printed unless you give the -\texttt{f} option, which prints the login name.

-\texttt{U uidlist} Lists information for processes whose real user ID numbers or login names are given in \texttt{uidlist}. The \texttt{uidlist} must be a single argument in the form of a blank- or comma-separated list.

-\texttt{y} Under a long listing (-\texttt{l}), omits the obsolete \texttt{F} and \texttt{ADDR} columns and includes an \texttt{RSS} column to report the resident set size of the process. Under the -\texttt{y} option, both \texttt{RSS} and \texttt{SZ} (see below) will be reported in units of kilobytes instead of pages.

Many of the options shown are used to select processes to list. If any are specified, the default list will be ignored and \texttt{ps} will select the processes represented by the inclusive OR of all the selection-criteria options.

Under the -\texttt{f} option, \texttt{ps} tries to determine the command name and arguments given when the process was created by examining the user block. Failing this, the command name is printed, as it would have appeared without the -\texttt{f} option, in square brackets.

The column headings and the meaning of the columns in a \texttt{ps} listing are given below; the letters \texttt{f} and \texttt{l} indicate the option (full or long, respectively) that causes the corresponding heading to appear; \texttt{all} means that the heading always appears. \texttt{Note:} These two options determine only what information is provided for a process; they do not determine which processes will be listed.

-\texttt{F (l)} Flags (hexadecimal and additive) associated with the process. These flags are available for historical purposes; no meaning should be currently ascribed to them.

-\texttt{S (l)} The state of the process:

  - \texttt{O} Process is running on a processor.
  - \texttt{S} Sleeping: process is waiting for an event to complete.
Runnable: process is on run queue.
Z Zombie state: process terminated and parent not waiting.
T Process is stopped, either by a job control signal or because it is being traced.

UID (f,l) The effective user ID number of the process (the login name is printed under the -f option).

PID (all) The process ID of the process (this datum is necessary in order to kill a process).

PPID (f,l) The process ID of the parent process.

C (f,l) Processor utilization for scheduling (obsolete). Not printed when the -c option is used.

CLS (f,l) Scheduling class. Printed only when the -c option is used.

PRI (l) The priority of the process. Without the -c option, higher numbers mean lower priority. With the -c option, higher numbers mean higher priority.

NI (l) Nice value, used in priority computation. Not printed when the -c option is used. Only processes in the certain scheduling classes have a nice value.

ADDR (l) The memory address of the process.

SZ (l) The total size of the process in virtual memory, including all mapped files and devices, in pages. See pagesize(1).

WCHAN (l) The address of an event for which the process is sleeping (if blank, the process is running).

STIME (f) The starting time of the process, given in hours, minutes, and seconds. (A process begun more than twenty-four hours before the ps inquiry is executed is given in months and days.)

TTY (all) The controlling terminal for the process (the message, ?, is printed when there is no controlling terminal).

TIME (all) The cumulative execution time for the process.

CMD (all) The command name (the full command name and its arguments, up to a limit of 80 characters, are printed under the -f option).

The following two additional columns are printed when the -j option is specified:

PGID The process ID of the process group leader.
SID The process ID of the session leader.

The following two additional columns are printed when the -L option is specified:
LWP       The lwp ID of the lwp being reported.
NLWP      The number of lwps in the process (if -f is also specified).

Under the -L option, one line is printed for each lwp in the process and the
time-reporting fields STIME and TIME show the values for the lwp, not the process. A
traditional single-threaded process contains only one lwp.

A process that has exited and has a parent, but has not yet been waited for by the
parent, is marked <defunct>.

-o format  The -o option allows the output format to be specified under user control.

The format specification must be a list of names presented as a single argument,
blank- or comma-separated. Each variable has a default header. The default header
can be overridden by appending an equals sign and the new text of the header. The
rest of the characters in the argument will be used as the header text. The fields
specified will be written in the order specified on the command line, and should be
arranged in columns in the output. The field widths will be selected by the system to
be at least as wide as the header text (default or overridden value). If the header text is
null, such as -o user=, the field width will be at least as wide as the default header
text. If all header text fields are null, no header line will be written.

The following names are recognized in the POSIX locale:

user       The effective user ID of the process. This will be the textual user
           ID, if it can be obtained and the field width permits, or a decimal
           representation otherwise.

ruser      The real user ID of the process. This will be the textual user ID, if it
can be obtained and the field width permits, or a decimal
representation otherwise.

group      The effective group ID of the process. This will be the textual
group ID, if it can be obtained and the field width permits, or a
decimal representation otherwise.

rgroup     The real group ID of the process. This will be the textual group ID,
            if it can be obtained and the field width permits, or a decimal
            representation otherwise.

pid        The decimal value of the process ID.

ppid       The decimal value of the parent process ID.

pgid       The decimal value of the process group ID.

pcpu       The ratio of CPU time used recently to CPU time available in the
           same period, expressed as a percentage. The meaning of “recently”
in this context is unspecified. The CPU time available is
determined in an unspecified manner.

vsz        The total size of the process in virtual memory, in kilobytes.
nice The decimal value of the system scheduling priority of the process.
See nice(1).

etime In the POSIX locale, the elapsed time since the process was started,
in the form:

\[ [ \text{dd-} \text{hh:} \text{mm:ss} \]

where
\[ \text{dd} \] is the number of days
\[ \text{hh} \] is the number of hours
\[ \text{mm} \] is the number of minutes
\[ \text{ss} \] is the number of seconds

The \text{dd} field will be a decimal integer. The \text{hh}, \text{mm} and \text{ss} fields will
be two-digit decimal integers padded on the left with zeros.

time In the POSIX locale, the cumulative CPU time of the process in the
form:

\[ [ \text{dd-} \text{hh:mm:ss} \]

The \text{dd}, \text{hh}, \text{mm}, and \text{ss} fields will be as described in the etime
specifier.

tty The name of the controlling terminal of the process (if any) in the
same format used by the who(1) command.

comm The name of the command being executed (argv[0] value) as a
string.

args The command with all its arguments as a string. The
implementation may truncate this value to the field width; it is
implementation-dependent whether any further truncation occurs.
It is unspecified whether the string represented is a version of the
argument list as it was passed to the command when it started, or
is a version of the arguments as they may have been modified by
the application. Applications cannot depend on being able to
modify their argument list and having that modification be
reflected in the output of \text{ps}. The Solaris implementation limits the
string to 80 bytes; the string is the version of the argument list as it
was passed to the command when it started.

The following names are recognized in the Solaris implementation:
f Flags (hexadecimal and additive) associated with the process.
s The state of the process.
c Processor utilization for scheduling (obsolete).
uid
The effective user ID number of the process as a decimal integer.
ru+id
The real user ID number of the process as a decimal integer.
gid
The effective group ID number of the process as a decimal integer.
rgid
The real group ID number of the process as a decimal integer.
projid
The project ID number of the process as a decimal integer.
project
The project ID of the process as a textual value if that value can be
obtained; otherwise as a decimal integer.
sid
The process ID of the session leader.
taskid
The task ID of the process.
class
The scheduling class of the process.
pri
The priority of the process. Higher numbers mean higher priority.
opri
The obsolete priority of the process. Lower numbers mean higher
priority.
lwp
The decimal value of the lwp ID. Requesting this formatting
option causes one line to be printed for each lwp in the process.
nlwp
The number of lwps in the process.
psr
The number of the processor to which the process or lwp is bound.
pset
The ID of the processor set to which the process or lwp is bound.
addr
The memory address of the process.
osz
The total size of the process in virtual memory, in pages.
wchan
The address of an event for which the process is sleeping (if −, the
process is running).
stime
The starting time or date of the process, printed with no blanks.
rss
The resident set size of the process, in kilobytes.
pmem
The ratio of the process’s resident set size to the physical memory
on the machine, expressed as a percentage.
fname
The first 8 bytes of the base name of the process’s executable file.

Only comm and args are allowed to contain blank characters; all others, including the
Solaris implementation variables, are not.

The following table specifies the default header to be used in the POSIX locale
corresponding to each format specifier.
The following table lists the Solaris implementation format specifiers and the default header used with each.

<table>
<thead>
<tr>
<th>Format Specifier</th>
<th>Default Header</th>
<th>Format Specifier</th>
<th>Default Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>args</td>
<td>COMMAND</td>
<td>ppid</td>
<td>PPID</td>
</tr>
<tr>
<td>comm</td>
<td>COMMAND</td>
<td>rgroup</td>
<td>RGROUP</td>
</tr>
<tr>
<td>etime</td>
<td>ELAPSED</td>
<td>ruser</td>
<td>RUSER</td>
</tr>
<tr>
<td>group</td>
<td>GROUP</td>
<td>time</td>
<td>TIME</td>
</tr>
<tr>
<td>nice</td>
<td>NI</td>
<td>tty</td>
<td>TT</td>
</tr>
<tr>
<td>pcpu</td>
<td>%CPU</td>
<td>user</td>
<td>USER</td>
</tr>
<tr>
<td>pgid</td>
<td>PGID</td>
<td>vsz</td>
<td>VSZ</td>
</tr>
<tr>
<td>pid</td>
<td>PID</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**EXAMPLE 1** An example of the `ps` command

The command:

```
example% ps -o user,pid,ppid=MOM -o args
```
EXAMPLE 1  An example of the ps command

writes the following in the POSIX locale:

```
USER   PID  MOM  COMMAND
helene 34   12   ps -o uid,pid,ppid=MOM -o args
```

The contents of the COMMAND field need not be the same due to possible truncation.

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of ps: LC_CTYPE, LC_MESSAGES, LC_TIME, and NLSPATH.

COLUMNS  Override the system-selected horizontal screen size, used to determine the number of text columns to display.

EXIT STATUS

The following exit values are returned:

- 0  Successful completion.
- >0  An error occurred.

FILES

- /dev/pts/*
- /dev/term/*  terminal ("tty") names searcher files
- /etc/passwd  UID information supplier
- /proc/*  process control files
- /tmp/ps_data  internal data structure

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled (see USAGE)</td>
</tr>
</tbody>
</table>

SEE ALSO

kill(1), nice(1), pagesize(1), pgrep(1), priocntl(1), who(1), getty(1M), proc(4), ttysrch(4), attributes(5), environ(5)

NOTES

Things can change while ps is running. The snapshot it gives is true only for a split-second, and it may not be accurate by the time you see it. Some data printed for defunct processes is irrelevant.

If no options to select processes are specified, ps will report all processes associated with the controlling terminal. If there is no controlling terminal, there will be no report other than the header.
ps -ef or ps -ostime may not report the actual start of a tty login session, but rather an earlier time, when a getty was last respawned on the tty line.

ps is CSI-enabled except for login names (usernames).
ps(1B)

NAME
ps – display the status of current processes

SYNOPSIS
/usr/ucb/ps [-acegnrSuUvwx] [-t term] [num]

DESCRIPTION
The ps command displays information about processes. Normally, only those processes that are running with your effective user ID and are attached to a controlling terminal (see termio(7I)) are shown. Additional categories of processes can be added to the display using various options. In particular, the -a option allows you to include processes that are not owned by you (that do not have your user ID), and the -x option allows you to include processes without controlling terminals. When you specify both -a and -x, you get processes owned by anyone, with or without a controlling terminal. The -r option restricts the list of processes printed to running and runnable processes.

ps displays in tabular form the process ID, under PID; the controlling terminal (if any), under TT; the cpu time used by the process so far, including both user and system time, under TIME; the state of the process, under S; and finally, an indication of the COMMAND that is running.

The state is given by a single letter from the following:

O  Process is running on a processor.
S  Sleeping. Process is waiting for an event to complete.
R  Runnable. Process is on run queue.
Z  Zombie state. Process terminated and parent not waiting.
T  Traced. Process stopped by a signal because parent is tracing it.

OPTIONS
The following options must all be combined to form the first argument:

- a  Includes information about processes owned by others.
- c  Displays the command name rather than the command arguments.
- e  Displays the environment as well as the arguments to the command.
- g  Displays all processes. Without this option, ps only prints interesting processes. Processes are deemed to be uninteresting if they are process group leaders. This normally eliminates top-level command interpreters and processes waiting for users to login on free terminals.
- l  Displays a long listing, with fields F, PPID, CP, PRI, NI, SZ, RSS, and WCHAN as described below.
- n  Produces numerical output for some fields. In a user listing, the USER field is replaced by a UID field.
- r  Restricts output to running and runnable processes.
- S  Displays accumulated CPU time used by this process and all of its reaped children.
Lists only process data associated with the terminal, `term`. Terminal identifiers may be specified in one of two forms: the device’s file name (for example, `tty04` or `term/14`) or, if the device’s file name starts with `tty`, just the digit identifier (for example, `04`).

Displays user-oriented output. This includes fields `USER`, `%CPU`, `%MEM`, `SZ`, `RSS`, and `START` as described below.

Updates a private database where `ps` keeps system information.

Displays a version of the output containing virtual memory. This includes fields `SIZE`, `%CPU`, `%MEM`, and `RSS`, described below.

Uses a wide output format (132 columns rather than 80); if repeated, that is, `-ww`, use arbitrarily wide output. This information is used to decide how much of long commands to print.

Includes processes with no controlling terminal.

A process number may be given, in which case the output is restricted to that process. This option must be supplied last.

Fields that are not common to all output formats:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>USER</code></td>
<td>Name of the owner of the process.</td>
</tr>
<tr>
<td><code>%CPU</code></td>
<td>CPU use of the process; this is a decaying average over up to a minute of previous (real) time.</td>
</tr>
<tr>
<td><code>NI</code></td>
<td>Process scheduling increment (see <code>getpriority(3C)</code> and <code>nice(3UCB)</code>).</td>
</tr>
<tr>
<td><code>SIZE</code></td>
<td>The total size of the process in virtual memory, including all mapped files and devices, in kilobyte units.</td>
</tr>
<tr>
<td><code>SZ</code></td>
<td>Same as <code>SIZE</code>.</td>
</tr>
<tr>
<td><code>RSS</code></td>
<td>Real memory (resident set) size of the process, in kilobyte units.</td>
</tr>
<tr>
<td><code>UID</code></td>
<td>Numerical user-ID of process owner.</td>
</tr>
<tr>
<td><code>PPID</code></td>
<td>Numerical ID of parent of process.</td>
</tr>
<tr>
<td><code>CP</code></td>
<td>Short-term CPU utilization factor (used in scheduling).</td>
</tr>
<tr>
<td><code>PRI</code></td>
<td>The priority of the process (higher numbers mean lower priority).</td>
</tr>
<tr>
<td><code>START</code></td>
<td>The starting time of the process, given in hours, minutes, and seconds. A process begun more than 24 hours before the <code>ps</code> inquiry is executed is given in months and days.</td>
</tr>
<tr>
<td><code>WCHAN</code></td>
<td>The address of an event for which the process is sleeping (if blank, the process is running).</td>
</tr>
<tr>
<td><code>%MEM</code></td>
<td>The ratio of the process’s resident set size to the physical memory on the machine, expressed as a percentage.</td>
</tr>
</tbody>
</table>
Flags (hexadecimal and additive) associated with the process. These flags are available for historical purposes; no meaning should be currently ascribed to them.

A process that has exited and has a parent, but has not yet been waited for by the parent, is marked `<defunct>`; otherwise, `ps` tries to determine the command name and arguments given when the process was created by examining the user block.

FILES

/dev
/dev/kmem kernel virtual memory
/dev/mem memory
/dev/swap default swap device
/dev/tty*
/etc/passwd UID information supplier

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

SEE ALSO

`kill(1), ps(1), whodo(1M), getpriority(3C), nice(3UCB), proc(4), attributes(5), termio(7I)`

NOTES

Things can change while `ps` is running; the picture it gives is only a close approximation to the current state. Some data printed for defunct processes is irrelevant.
NAME  pvs – display the internal version information of dynamic objects

SYNOPSIS  pvs [-Cdlnosv] [-N name] file...

DESCRIPTION  The pvs utility displays any internal version information contained within an ELF file. Commonly, these files are dynamic executables and shared objects, and possibly relocatable objects. This version information can fall into one of two categories:

- version definitions
- version dependencies

Version definitions describe the interfaces made available by an ELF file. Each version definition is associated to a set of global symbols provided by the file. Version definitions may be assigned to a file during its creation by the link-editor using the -M option and the associated mapfile directives (see the Linker and Libraries Guide for more details).

Version dependencies describe the binding requirements of dynamic objects on the version definitions of any shared object dependencies. When a dynamic object is built with a shared object, the link-editor records information within the dynamic object indicating that the shared object is a dependency. This dependency must be satisfied at runtime. If the shared object also contains version definitions, then those version definitions that satisfy the global symbol requirements of the dynamic object will also be recorded in the dynamic object being created. At process initialization, the runtime linker will use any version dependencies as a means of validating the interface requirements of the dynamic objects used to construct the process.

OPTIONS  The following options are supported. If neither the -d or -r options are specified, both will be enabled.

- **C**  Demangles C++ symbol names.
- **d**  Prints version definition information.
- **l**  When used with the -s option, prints any symbols that have been reduced from global to local binding due to versioning. By convention, these symbol entries are located in the .symtab section, and fall between the FILE symbol representing the output file, and the FILE symbol representing the first input file used to generate the output file. These reduced symbol entries are assigned the fabricated version definition _REDUCED_. No reduced symbols will be printed if the file has been stripped (see strip(1)), or if the symbol entry convention cannot be determined.
- **n**  Normalizes version definition information. By default, all version definitions within the object are displayed. However, version definitions may inherit other version definitions, and under normalization only the head of each inheritance list is displayed.
- **N name**  Prints only the information for the given version definition name and any of its inherited version definitions (when used with the -d option), or for the given dependency file name (when used with the -r option).
pvs(1)

-\(o\)
Creates one-line version definition output. By default, file, version definitions, and any symbol output is indented to ease human inspection. This option prefixes each output line with the file and version definition name and may be more useful for analysis with automated tools.

-\(r\)
Prints version dependency (requirements) information.

-\(s\)
Prints the symbols associated with each version definition. Any data symbols are accompanied with the size, in bytes, of the data item.

-\(v\)
Verbose output. Indicates any weak version definitions, and any version definition inheritance. When used with the -\(N\) and -\(d\) options, the inheritance of the base version definition is also shown. When used with the -\(s\) option, the version symbol definition is also shown.

**OPERANDS**

The following operands are supported.

*file*  
The ELF file about which internal version information is displayed.

**EXAMPLES**

**EXAMPLE 1** Displaying version definitions

The following example displays the version definitions of *libelf.so.1*:

\(\%\) pvs -d /usr/lib/libelf.so.1

libelf.so.1;
SUNW_1.1

**EXAMPLE 2** Creating a one-liner display

A normalized, one-liner display, suitable for creating a *mapfile* version control directive, can be created using the -\(n\) and -\(o\) options:

\(\%\) pvs -don /usr/lib/libelf.so.1

/usr/lib/libelf.so.1 - SUNW_1.1;

**EXAMPLE 3** Displaying version requirements

The following example displays the version requirements of *ldd* and *pvs*:

\(\%\) pvs -r /usr/bin/ldd /usr/bin/pvs

/usr/bin/ldd:
  libelf.so.1 (SUNW_1.1);
  libc.so.1 (SUNW_1.1);
/usr/bin/pvs:
  libelf.so.1 (SUNW_1.1);
  libc.so.1 (SUNW_1.1);

**EXIT STATUS**

If the requested version information is not found, a non-zero value is returned; otherwise a 0 value is returned.

Version information is determined not found when any of the following is true:

- the -\(d\) option is specified and no version definitions are found;
the -r option is specified and no version requirements are found;
- neither the -d nor -r option is specified and no version definitions or version requirements are found.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWtoo</td>
</tr>
</tbody>
</table>

SEE ALSO

ld(1), ldd(1), strip(1), elf(3ELF), attributes(5)

Linker and Libraries Guide
pwd(1)

NAME    pwd – return working directory name
SYNOPSIS /usr/bin/pwd
DESCRIPTION pwd writes an absolute path name of the current working directory to standard output.

Both the Bourne shell, sh(1), and the Korn shell, ksh(1), also have a built-in pwd command.

ENVIRONMENT VARIABLES See environ(5) for descriptions of the following environment variables that affect the execution of pwd: LC_MESSAGES and NLSPATH.

EXIT STATUS The following exit values are returned:
0    Successful completion.
>0    An error occurred.

If an error is detected, output will not be written to standard output, a diagnostic message will be written to standard error, and the exit status will not be 0.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

SEE ALSO cd(1), ksh(1), sh(1), shell_builtins(1), attributes(5), environ(5)

DIAGNOSTICS “Cannot open ..” and “Read error in ..” indicate possible file system trouble and should be referred to a UNIX system administrator.

NOTES If you move the current directory or one above it, pwd may not give the correct response. Use the cd(1) command with a full path name to correct this situation.
NAME | ranlib – convert archives to random libraries

SYNOPSIS | /usr/ccs/bin/ranlib archive

DESCRIPTION | The ranlib utility was used in SunOS 4.x to add a table of contents to archive libraries, which converted each archive to a form that could be linked more rapidly. This is no longer needed, as the ar(1) command automatically provides all the functionality ranlib used to provide.

This script is provided as a convenience for software developers who need to maintain Makefiles that are portable across a variety of operating systems.

EXIT STATUS | ranlib has exit status 0.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWbtool</td>
</tr>
</tbody>
</table>

SEE ALSO | ar(1), ar(3HEAD), attributes(5)
NAME
rcp - remote file copy

SYNOPSIS
rcp [-p] filename1 filename2
rcp [-pr] filename... directory

DESCRIPTION
The rcp command copies files between machines. Each filename or directory argument
is either a remote file name of the form:
hostname:pathname or a local file name (containing no ":" (colon) characters, or "/" (backslash)
before any ":" (colon) characters).

The hostname can be an IPv4 or IPv6 address string. See inet(7P) and inet6(7P).
Since IPv6 addresses already contain colons, the hostname should be enclosed in a pair
of square brackets when an IPv6 address is used. Otherwise, the first occurrence of a
colon can be interpreted as the separator between hostname and path. For example,
[1080::8:800:200C:417A]:tmp/file

If a filename is not a full path name, it is interpreted relative to your home directory on
hostname. A path on a remote host may be quoted using \", " , or ' , so that the
metacharacters are interpreted remotely.

rcp does not prompt for passwords; your current local user name must exist on
hostname and allow remote command execution by rsh(1).

rcp handles third party copies, where neither source nor target files are on the current
machine. Hostnames may also take the form
username@hostname:filename to use username rather than your current local user name as the
user name on the remote host. rcp also supports Internet domain addressing of the
remote host, so that:
username@hostname.domain:filename specifies the username to be used, the hostname, and the
domain in which that host resides. File names that are not full path names will be
interpreted relative to the home directory of the user named username, on the remote
host.

OPTIONS
The following options are supported:
-p     Attempts to give each copy the same modification times, access times,
       modes, and ACLs if applicable as the original file.
-r     Copies each subtree rooted at filename; in this case the destination must be a
directory.

USAGE
See largefile(5) for the description of the behavior of rcp when encountering files
greater than or equal to 2 Gbyte (2^31 bytes).

The rcp command is IPv6–enabled. See ip6(7P).
rcp(1)

FILES

$HOME/.profile

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWrcmdc</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

SEE ALSO

cpio(1), ftp(1), rlogin(1), rsh(1), setfacl(1), tar(1), hosts.equiv(4),
attributes(5), largefile(5), inet(7P), inet6(7P), ip6(7P)

NOTES

rcp is meant to copy between different hosts; attempting to rcp a file onto itself, as
with:

example% rcp tmp/file myhost:/tmp/file

results in a severely corrupted file.

rcp may not correctly fail when the target of a copy is a file instead of a directory.

rcp can become confused by output generated by commands in a $HOME/.profile
on the remote host.

rcp requires that the source host have permission to execute commands on the remote
host when doing third-party copies.

rcp does not properly handle symbolic links. Use tar or cpio piped to rsh to obtain
remote copies of directories containing symbolic links or named pipes. See tar(1) and
cpio(1).

If you forget to quote metacharacters intended for the remote host, you will get an
incomprehensible error message.

rcp will fail if you copy ACLs to a file system that does not support ACLs.

rcp is CSI-enabled except for the handling of username, hostname, and domain.
rdist(1)

NAME

rdist – remote file distribution program

SYNOPSIS


[login @] hostname [ : destpath]

DESCRIPTION

The rdist utility maintains copies of files on multiple hosts. It preserves the owner, group, mode, and modification time of the master copies, and can update programs that are executing. (Note: rdist does not propagate ownership or mode changes when the file contents have not changed.) Normally, a copy on a remote host is updated if its size or modification time differs from the original on the local host. (With the -y option (younger mode), only the modification times are checked, not the size. See OPTIONS below.)

There are two forms of the rdist command. In the first form shown in the SYNOPSIS section above, rdist reads the indicated distfile for instructions on updating files and/or directories. If distfile is ‘-’, the standard input is used. If no -f option is present, rdist first looks in its working directory for distfile, and then for Distfile, for instructions.

The second form shown in SYNOPSIS uses the -c option and specifies paths as command line options.

In order to be able to use rdist across machines, each host machine must have a /etc/host.equiv file, or the user must have an entry in the .rhosts file in the home directory. See hosts.equiv(4) for more information.

OPTIONS

The following options are supported:

-b

Binary comparison. Performs a binary comparison and updates files if they differ, rather than merely comparing dates and sizes.

-c pathname ... [login @] hostname [ : destpath ]

Copies each pathname to the named host; if destpath is specified, it will not update any pathname on the named host. (Relative filenames are taken as relative to your home directory.) If the ‘login @’ prefix is given, the update is performed with the user ID of login. If the ‘: destpath’ is given, the remote file is installed as that pathname.

-a macro=value

Defines macro to have value. This option is used to define or override macro definitions in the distfile. value can be the empty string, one name, or a list of names surrounded by parentheses and separated by white space.

-D

Enables debugging.
-f distfile
  Uses the description file distfile. A ‘-’ as the distfile argument denotes the standard input.

-h
  Follows symbolic links. Copies the file that the link points to rather than the link itself.

-i
  Ignores unresolved links. rdist will normally try to maintain the link structure of files being transferred and warn the user if all the links cannot be found.

-m host
  Limits which machines are to be updated. Multiple -m arguments can be given to limit updates to a subset of the hosts listed in the distfile.

-n
  Prints the commands without executing them. This option is useful for debugging a distfile.

-q
  Quiet mode. Does not display the files being updated on the standard output.

-R
  Removes extraneous files. If a directory is being updated, removes files on the remote host that do not correspond to those in the master (local) directory. This is useful for maintaining truly identical copies of directories.

-v
  Verifies that the files are up to date on all the hosts. Any files that are out of date are displayed, but no files are updated, nor is any mail sent.

-w
  Whole mode. The whole file name is appended to the destination directory name. Normally, only the last component of a name is used when renaming files. This preserves the directory structure of the files being copied, instead of flattening the directory structure. For instance, renaming a list of files such as dir1/dir2 to dir3 would create files dir3/dir1 and dir3/dir2 instead of dir3 and dir3.
  When the -w option is used with a filename that begins with ~, everything except the home directory is appended to the destination name.

-y
  Younger mode. Does not update remote copies that are younger than the master copy, but issues a warning message instead. Only modification times are checked. No comparison of size is made.

White Space Characters
NEWLINE, TAB, and SPACE characters are all treated as white space; a mapping continues across input lines until the start of the next mapping: either a single filename followed by a ‘- >’ or the opening parenthesis of a filename list.

Comments
Comments begin with # and end with a NEWLINE.
The distfile contains a sequence of entries that specify the files to be copied, the destination files to be copied, the destination hosts, and what operations to perform to do the updating. Each entry has one of the following formats:

```
variable_name '==' name_list
[ label: ] source_list '->' destination_list command_list
[ label: ] source_list ':::' time_stamp file command_list
```

The first format is used for defining variables. The second format is used for distributing files to other hosts. The third format is used for making lists of files that have been changed since some given date. The source list specifies a list of files and/or directories on the local host that are to be used as the master copy for distribution. The destination list is the list of hosts to which these files are to be copied. Each file in the source list is added to a list of changes if the file is out of date on the host that is being updated (second format) or if the file is newer than the time stamp file (third format). Labels are optional. They are used to identify a command for partial updates. The colon (:) is used after an optional label, while the double colon (::) is used for making lists of files that have been changed since a certain date (specified by the date/time of the time_stamp file). Typically, only `notify` is used with the `::` format of the command line.

**Macros**

`rdist` has a limited macro facility. Macros are only expanded in filename or hostname lists, and in the argument lists of certain primitives. Macros cannot be used to stand for primitives or their options, or the `-'` or `'::'` symbols.

A macro definition is a line of the form:

```
macro = value
```

A macro reference is a string of the form:

```
${macro}
```

although (as with `make(1S)`) the braces can be omitted if the macro name consists of just one character.

**Metacharacters**

The shell meta-characters: `[`, `]`, `{`, `}`, `*` and `?` are recognized and expanded (on the local host only) just as they are with `csh(1)`. Metacharacters can be escaped by prepending a backslash.

The `~` character is also expanded in the same way as with `csh`; however, it is expanded separately on the local and destination hosts.

**Filenames**

File names that do not begin with `' '/'` or ``~` are taken to be relative to user’s home directory on each destination host; they are not relative to the current working directory. Multiple file names must be enclosed within parentheses.

**Primitives**

The following primitives can be used to specify actions `rdist` is to take when updating remote copies of each file.
    Copy out of date files and directories (recursively). If no newname operand is given, the name of the local file is given to the remote host’s copy. If absent from the remote host, parent directories in a filename’s path are created. To help prevent disasters, a non-empty directory on a target host is not replaced with a regular file or a symbolic link by rdist. However, when using the -R option, a non-empty directory is removed if the corresponding filename is completely absent on the master host.

The options for install have the same semantics as their command line counterparts, but are limited in scope to a particular map. The login name used on the destination host is the same as the local host unless the destination name is of the format login@host. In that case, the update is performed under the username login.

notify address . . .
    Send mail to the indicated email address of the form:

    user@host

    that lists the files updated and any errors that may have occurred. If an address does not contain a '@host' suffix, rdist uses the name of the destination host to complete the address.

except filename . . .
    Omit from updates the files named as arguments.

except _pat pattern . . .
    Omit from updates the filenames that match each regular-expression pattern (see ed(1) for more information on regular expressions). Note that '\', and '$' characters must be escaped in the distfile. Shell variables can also be used within a pattern, however shell filename expansion is not supported.

special [filename] . . . "command-line"
    Specify a Bourne shell, sh(1) command line to execute on the remote host after each named file is updated. If no filename argument is present, the command-line is performed for every updated file, with the shell variable FILE set to the file’s name on the local host. The quotation marks allow command-line to span input lines in the distfile; multiple shell commands must be separated by semicolons (;).

    The default working directory for the shell executing each command-line is the user’s home directory on the remote host.

IPv6
    The rdist command is IPv6-enabled. See ip6(7P).

EXAMPLES

EXAMPLE 1 A sample distfile
    The following sample distfile instructs rdist to maintain identical copies of a shared library, a shared-library initialized data file, several include files, and a directory, on hosts named hermes and magus. On magus, commands are executed as super-user.
EXAMPLE 1 A sample distfile (Continued)

rdist notifies merlin@druid whenever it discovers that a local file has changed relative to a timestamp file. (Parentheses are used when the source or destination list contains zero or more names separated by white-space.)

HOSTS = ( hermes root@magus )

FILES = ( /usr/local/lib/libcant.so.1.1
   /usr/local/lib/libcant.sa.1.1 /usr/local/include/{*.h}
   /usr/local/bin )

(${FILES}) -> (${HOSTS})

install -R;

$({FILES}) :: /usr/local/lib/timestamp
   notify merlin@druid ;

FILES

-/rhosts user’s trusted hosts and users
/etc/host.equiv system trusted hosts and users
/tmp/rdist* temporary file for update lists

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWrcmdc</td>
</tr>
</tbody>
</table>

SEE ALSO

csh(1), ed(1), make(1S), sh(1), stat(2), hosts.equiv(4), attributes(5), ip6(7P)

DIAGNOSTICS

A complaint about mismatch of rdist version numbers may really stem from some problem with starting your shell, for example, you are in too many groups.

WARNINGS

The super-user does not have its accustomed access privileges on NFS mounted file systems. Using rdist to copy to such a file system may fail, or the copies may be owned by user “nobody”.

BUGS

Source files must reside or be mounted on the local host.

There is no easy way to have a special command executed only once after all files in a directory have been updated.

Variable expansion only works for name lists; there should be a general macro facility.

rdist aborts on files that have a negative modification time (before Jan 1, 1970).

There should be a “force” option to allow replacement of non-empty directories by regular files or symlinks. A means of updating file modes and owners of otherwise identical files is also needed.
The read utility will read a single line from standard input.

By default, unless the -r option is specified, backslash (\) acts as an escape character. If standard input is a terminal device and the invoking shell is interactive, read will prompt for a continuation line when:

- The shell reads an input line ending with a backslash, unless the -r option is specified.
- A here-document is not terminated after a newline character is entered.

The line will be split into fields as in the shell; the first field will be assigned to the first variable var, the second field to the second variable var, and so forth. If there are fewer var operands specified than there are fields, the leftover fields and their intervening separators will be assigned to the last var. If there are fewer fields than vars, the remaining vars will be set to empty strings.

The setting of variables specified by the var operands will affect the current shell execution environment. If it is called in a subshell or separate utility execution environment, such as one of the following:

```
(read foo)
noglob read ...
find . -exec read ... 
```

it will not affect the shell variables in the caller’s environment.

The standard input must be a text file.

In the sh shell:

One line is read from the standard input and, using the internal field separator, IFS (normally space or tab), to delimit word boundaries, the first word is assigned to the first name, the second word to the second name, etc., with leftover words assigned to the last name. Lines can be continued using \newline. Characters other than newline can be quoted by preceding them with a backslash. These backslashes are removed before words are assigned to names, and no interpretation is done on the character that follows the backslash. The return code is 0, unless an EOF is encountered.

In the csh shell:

The notation:

```
set variable = $<
```

loads one line of standard input as the value for variable. (See csh(1)).
The shell input mechanism. One line is read and is broken up into fields using the characters in \texttt{IFS} as separators. The escape character, (\texttt{\)}, is used to remove any special meaning for the next character and for line continuation. In raw mode, -r, the \ character is not treated specially. The first field is assigned to the first \texttt{name}, the second field to the second \texttt{name}, etc., with leftover fields assigned to the last \texttt{name}. The \texttt{-p} option causes the input line to be taken from the input pipe of a process spawned by the shell using \texttt{|&}. If the -s flag is present, the input will be saved as a command in the history file. The flag -u can be used to specify a one digit file descriptor unit \texttt{n} to read from. The file descriptor can be opened with the \texttt{exec} special command. The default value of \texttt{n} is 0. If \texttt{name} is omitted then \texttt{REPLY} is used as the default \texttt{name}. The exit status is 0 unless the input file is not open for reading or an end-of-file is encountered. An end-of-file with the \texttt{-p} option causes cleanup for this process so that another can be spawned. If the first argument contains a \?, the remainder of this word is used as a \texttt{prompt} on standard error when the shell is interactive. The exit status is 0 unless an end-of-file is encountered.

**OPTIONS**

The following option is supported:

- \texttt{-r} Do not treat a backslash character in any special way. Consider each backslash to be part of the input line.

**OPERANDS**

The following operand is supported:

\texttt{var} The name of an existing or non-existing shell variable.

**EXAMPLES**

**EXAMPLE 1** An example of the \texttt{read} command.

The following example for \texttt{/usr/bin/read} prints a file with the first field of each line moved to the end of the line.

```bash
while read -r xx yy
do
  printf "\%s $s\n" "$yy" "$xx"
done < input_file
```

**ENVIRONMENT VARIABLES**

See \texttt{environ(5)} for descriptions of the following environment variables that affect the execution of \texttt{read}: \texttt{LC_CTYPE}, \texttt{LC_MESSAGES}, and \texttt{NLSPATH}.

\texttt{IFS} Determine the internal field separators used to delimit fields.

\texttt{PS2} Provide the prompt string that an interactive shell will write to standard error when a line ending with a backslash is read and the -r option was not specified, or if a here-document is not terminated after a newline character is entered.

**EXIT STATUS**

The following exit values are returned:

- 0 Successful completion.
- >0 End-of-file was detected or an error occurred.

**ATTRIBUTES**

See \texttt{attributes(5)} for descriptions of the following attributes:
<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO  csh(1), ksh(1), line(1), set(1), sh(1), attributes(5), environ(5)
readfile(1F)

NAME
readfile, longline – reads file, gets longest line

SYNOPSIS
readfile filename
longline [filename]

DESCRIPTION
The readfile function reads filename and copies it to stdout. No translation of
NEWLINE is done. It keeps track of the longest line it reads and if there is a
subsequent call to longline, the length of that line, including the NEWLINE
character, is returned.

The longline function returns the length, including the NEWLINE character, of the
longest line in filename. If filename is not specified, it uses the file named in the last call
to readfile.

EXAMPLES
EXAMPLE 1 Typical use of readfile and longline

Here is a typical use of readfile and longline in a text frame definition file:

    ..
    ..
    text="'readfile myfile'"
columns='longline'
    ..
    ..

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO
cat(1), attributes(5)

DIAGNOSTICS
If filename does not exist, readfile will return FALSE (that is, the expression will
have an error return).

longline returns 0 if a readfile has not previously been issued.

NOTES
More than one descriptor can call readfile in the same frame definition file. In text
frames, if one of those calls is made from the text descriptor, then a subsequent use
of longline will always get the longest line of the file read by the readfile
associated with the text descriptor, even if it was not the most recent use of
readfile.
NAME
readonly – shell built-in function to protect the value of the given variable from reassignment

sh  readonly [name...]
ksh **readonly [name [= value] ...]

sh  The given names are marked readonly and the values of the these names may not be changed by subsequent assignment. If no arguments are given, a list of all readonly names is printed.
ksh  The given names are marked readonly and these names cannot be changed by subsequent assignment.

On this man page, ksh(1) commands that are preceded by one or two ** (asterisks) are treated specially in the following ways:

1. Variable assignment lists preceding the command remain in effect when the command completes.
2. I/O redirections are processed after variable assignments.
3. Errors cause a script that contains them to abort.
4. Words, following a command preceded by ** that are in the format of a variable assignment, are expanded with the same rules as a variable assignment. This means that tilde substitution is performed after the = sign and word splitting and file name generation are not performed.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO  ksh(1), sh(1), typeset(1), attributes(5)
refer(1)

NAME    refer – expand and insert references from a bibliographic database


DESCRIPTION refer is a preprocessor for nroff(1), or troff(1), that finds and formats references. The input files (standard input by default) are copied to the standard output, except for lines between ‘.’ [‘ and ‘.’] command lines. Such lines are assumed to contain keywords as for lookbib(1), and are replaced by information from a bibliographic data base. The user can avoid the search, override fields from it, or add new fields. The reference data, from whatever source, is assigned to a set of troff strings. Macro packages such as ms(5) print the finished reference text from these strings. A flag is placed in the text at the point of reference. By default, the references are indicated by numbers.

When refer is used with eqn(1), neqn, or tbl(1), refer should be used first in the sequence, to minimize the volume of data passed through pipes.

OPTIONS

-b      Bare mode — do not put any flags in text (neither numbers or labels).
-e      Accumulate references instead of leaving the references where encountered, until a sequence of the form:

   .[
   $LIST$
   .]

is encountered, and then write out all references collected so far. Collapse references to the same source.

-n      Do not search the default file.

-ar     Reverse the first r author names (Jones, J. A. instead of J. A. Jones). If r is omitted, all author names are reversed.

-cstring Capitalize (with SMALL CAPS) the fields whose key-letters are in string.

-kx     Instead of numbering references, use labels as specified in a reference data line beginning with the characters %x; By default, x is L.

-lm,n   Instead of numbering references, use labels from the senior author’s last name and the year of publication. Only the first m letters of the last name and the last n digits of the date are used. If either of m or n is omitted, the entire name or date, respectively, is used.

-p filename Take the next argument as a file of references to be searched. The default file is searched last.
-skeys

Sort references by fields whose key-letters are in the keys string, and permute reference numbers in the text accordingly. Using this option implies the -e option. The key-letters in keys may be followed by a number indicating how many such fields are used, with a + sign taken as a very large number. The default is AD, which sorts on the senior author and date. To sort on all authors and then the date, for instance, use the options ‘-sA+T’.

FILES
/usr/lib/refer directory of programs
/usr/lib/refer/papers directory of default publication lists and indexes

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWdoc</td>
</tr>
</tbody>
</table>

SEE ALSO addbib(1), eqn(1), indexbib(1), lookbib(1), nroff(1), roffbib(1), sortbib(1), tbl(1), troff(1), attributes(5)
The `regcmp` command performs a function similar to `regcmp` and, in most cases, precludes the need for calling `regcmp` from C programs. Bypassing `regcmp` saves on both execution time and program size. The command `regcmp` compiles the regular expressions in `filename` and places the output in `filename.i`.

If the – option is used, the output is placed in `filename.c`. The format of entries in `filename` is a name (C variable) followed by one or more blanks followed by one or more regular expressions enclosed in double quotes. The output of `regcmp` is C source code. Compiled regular expressions are represented as `extern char vectors`. `filename.i` files may thus be included in C programs, or `filename.c` files may be compiled and later loaded. In the C program that uses the `regcmp` output, `regex(abc,line)` applies the regular expression named `abc` to `line`. Diagnostics are self-explanatory.

The three arguments to `telno` shown above must all be entered on one line.

In the C program that uses the `regcmp` output,

`regex(telno, line, area, exch, rest)` applies the regular expression named `telno` to `line`.

A general description of the usage of the `LC_*` environmental variables can be found in `environ(5)`.

`LC_CTYPE` Determines how `regcmp` handles characters. When `LC_CTYPE` is set to a valid value, `regcmp` can display and handle text and filenames containing valid characters for that locale.

`LC_MESSAGES` Determines how diagnostic and informative messages are presented. This includes the language and style of the messages, and the correct form of affirmative and negative responses. In the "C" locale, the messages are presented in the default form found in the program itself (in most cases, U.S. English).

See `attributes(5)` for descriptions of the following attributes:
<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWtoo</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

SEE ALSO regcmp(3C), attributes(5), environ(5)
regex(1F)

NAME regex – match patterns against a string

SYNOPSIS regex [-e] [-v "string"] [pattern template] ... pattern [template]

DESCRIPTION The regex command takes a string from the standard input, and a list of pattern / template pairs, and runs regex() to compare the string against each pattern until there is a match. When a match occurs, regex writes the corresponding template to the standard output and returns TRUE. The last (or only) pattern does not need a template. If that is the pattern that matches the string, the function simply returns TRUE. If no match is found, regex returns FALSE.

The argument pattern is a regular expression of the form described in regex(). In most cases, pattern should be enclosed in single quotes to turn off special meanings of characters. Note that only the final pattern in the list may lack a template.

The argument template may contain the strings $m0 through $m9, which will be expanded to the part of pattern enclosed in ( . . . ) $0 through ( . . . ) $9 constructs (see examples below). Note that if you use this feature, you must be sure to enclose template in single quotes so that FMLI does not expand $m0 through $m9 at parse time. This feature gives regex much of the power of cut(1), paste(1), and grep(1), and some of the capabilities of sed(1). If there is no template, the default is $m0$m1$m2$m3$m4$m5$m6$m7$m8$m9.

OPTIONS The following options are supported:

- e Evaluates the corresponding template and writes the result to the standard output.

- v "string" Uses string instead of the standard input to match against patterns.

EXAMPLES

EXAMPLE 1 Cutting letters out of a string

To cut the 4th through 8th letters out of a string (this example will output strin and return TRUE):

`regex -v "my string is nice" ^.{3}(.{5})$0 $m0`

EXAMPLE 2 Validating input in a form

In a form, to validate input to field 5 as an integer:

`valid=`regex -v "$5" ^[0-9]+$`

EXAMPLE 3 Translating an environment variable in a form

In a form, to translate an environment variable which contains one of the numbers 1, 2, 3, 4, 5 to the letters a, b, c, d, e:

`value=`regex -v "$VAR1" [12345] a b c d e . * Error`

Note the use of the pattern ‘. *’ to mean "anything else".
EXAMPLE 4 Using backquoted expressions

In the example below, all three lines constitute a single backquoted expression. This expression, by itself, could be put in a menu definition file. Since backquoted expressions are expanded as they are parsed, and output from a backquoted expression (the `cat` command, in this example) becomes part of the definition file being parsed, this expression would read `/etc/passwd` and make a dynamic menu of all the login ids on the system.

```
'cat /etc/passwd | regex '"^[^[:;]*$0.*$ '"'
name=$m0
action='message "$m0 is a user\"''
```

DIAGNOSTICS

If none of the patterns match, `regex` returns `FALSE`, otherwise `TRUE`.

NOTES

Patterns and templates must often be enclosed in single quotes to turn off the special meanings of characters. Especially if you use the `$m0` through `$m9` variables in the template, since FMLI will expand the variables (usually to "") before `regex` even sees them.

Single characters in character classes (inside `[ ]`) must be listed before character ranges, otherwise they will not be recognized. For example, `[a-zA-Z_/>]` will not find underscores (`_`) or slashes (`/`), but `[/_a-zA-Z]` will.

The regular expressions accepted by `regcmp` differ slightly from other utilities (that is, `sed`, `grep`, `awk`, `ed`, and so forth).

`regex` with the `-e` option forces subsequent commands to be ignored. In other words, if a backquoted statement appears as follows:

```
'regex -e ...; command1; command2'
```

`command1` and `command2` would never be executed. However, dividing the expression into two:

```
'regex -e ...;\''command1; command2\''
```

would yield the desired result.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO: `awk(1), cut(1), grep(1), paste(1), sed(1), regcmp(3C), attributes(5)`
NAME  
reinit – runs an initialization file

SYNOPSIS  
reinit filename

DESCRIPTION  
The reinit command is used to change the values of descriptors defined in the initialization file that was named when fmli was invoked and/or define additional descriptors. FMLI will parse and evaluate the descriptors in filename, and then continue running the current application. The argument filename must be the name of a valid FMLI initialization file.

The reinit command does not re-display the introductory frame or change the layout of screen labels for function keys.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO  
attributes(5)
renice – alter priority of running processes

renice
[-n increment] [-i idtype] ID...

renice [-n increment] [-g | -p | -u]ID...

renice priority [-p] pid... [-g gid...] [-p pid...] [-u user...]

renice priority -g gid... [-g gid...] [-p pid...] [-u user...]

renice priority -u user... [-g gid...] [-p pid...] [-u user...]

DESCRIPTION

The renice command alters the scheduling priority of one or more running processes. By default, the processes to be affected are specified by their process IDs. If the first operand is a number within the valid range of priorities (−20 to 20), renice will treat it as a priority (as in all but the first synopsis form); otherwise, renice will treat it as an ID (as in the first synopsis form).

Altering Process Priority

Users other than the privileged user may only alter the priority of processes they own, and can only monotonically increase their “nice value” within the range 0 to 19. This prevents overriding administrative fiats. The privileged user may alter the priority of any process and set the priority to any value in the range −20 to 19. Useful priorities are: 19 (the affected processes will run only when nothing else in the system wants to); 0 (the “base” scheduling priority); and any negative value (to make things go very fast). 20 is an acceptable nice value, but will be rounded down to 19.

OPTIONS

renice supports the following option features:

- The first operand, priority, must precede the options and can have the appearance of a multi-digit option.
- The -g, -p, and -u options can each take multiple option-arguments.
- The pid option-argument can be used without its -p option.
- The -i option can be used to specify the ID type for the ID list. This is preferred in specifying ID type over the use of the -g | -p | -u syntax, which is now obsolete. See NOTES.

The following options are supported:

- g
  Interprets all operands or just the gid arguments as unsigned decimal integer process group IDs.

- i
  This option, together with the ID list arguments, specifies a class of processes to which the renice command is to apply. The interpretation of the ID list depends on the value of idtype. The valid idtype arguments are: pid, pgid, uid, gid, sid, taskid, and projid.

- n increment
  Specifies how the system scheduling priority of the specified process or processes is to be adjusted. The increment option-argument is a positive or negative decimal integer that will be used to modify the system scheduling priority of the specified
process or processes. Positive increment values cause a lower system scheduling priority. Negative increment values may require appropriate privileges and will cause a higher system scheduling priority.

-\textit{p}\hspace{1cm}
Interprets all operands or just the \textit{pid} arguments as unsigned decimal integer process IDs. The -\textit{p} option is the default if no options are specified.

-\textit{u}\hspace{1cm}
Interprets all operands or just the \textit{user} argument as users. If a user exists with a user name equal to the operand, then the user ID of that user will be used in further processing. Otherwise, if the operand represents an unsigned decimal integer, it will be used as the numeric user ID of the user.

**OPERANDS**

The following operands are supported:

\textit{ID}\hspace{1cm} A process ID, process group ID or user name/user ID, depending on the option selected.

\textit{priority}\hspace{1cm} The value specified is taken as the actual system scheduling priority, rather than as an increment to the existing system scheduling priority. Specifying a scheduling priority higher than that of the existing process may require appropriate privileges.

**EXAMPLES**

**EXAMPLE 1** Adjusting the scheduling priority of process IDs

Adjust the system scheduling priority so that process IDs 987 and 32 would have a lower scheduling priority:

```
example% renice -n 5 -p 987 32
```

**EXAMPLE 2** Adjusting the scheduling priority of group IDs

Adjust the system scheduling priority so that group IDs 324 and 76 would have a higher scheduling priority, if the user has the appropriate privileges to do so:

```
example% renice -n -4 -g 324 76
```

**EXAMPLE 3** Adjusting the scheduling priority of a user ID and user name

Adjust the system scheduling priority so that numeric user ID 8 and user \textit{sas} would have a lower scheduling priority:

```
example% renice -n 4 -u 8 sas
```

**ENVIRONMENT VARIABLES**

See \texttt{environ(5)} for descriptions of the following environment variables that affect the execution of \texttt{renice}: \texttt{LC\_CTYPE}, \texttt{LC\_MESSAGES}, and \texttt{NLSPATH}.

**EXIT STATUS**

The following exit values are returned:
Successful completion.

>0  An error occurred.

FILES

/etc/passwd  map user names to user IDs

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO

nice(1), passwd(1), priocntl(1), attributes(5), environ(5)

NOTES

The renice syntax

renice [-n increment] [-i idtype] ID ... is preferred over the old syntax

renice [-n increment] [-g | -p | -u] ID ... which is now obsolete.

If you make the priority very negative, then the process cannot be interrupted.

To regain control you must make the priority greater than 0.

Users other than the privileged user cannot increase scheduling priorities of their own processes, even if they were the ones that decreased the priorities in the first place.

The priocntl command subsumes the function of renice.
reset(1F)

NAME     reset – reset the current form field to its default values

SYNOPSIS     reset

DESCRIPTION     The reset function changes the entry in a field of a form to its default value; that is, the value displayed when the form was opened.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO     attributes(5)
**NAME**  
`rlogin` – remote login

**SYNOPSIS**  
```
rlogin [ -8EL ] [-ec ] [-l username] hostname
```

**DESCRIPTION**  
`rlogin` establishes a remote login session from your terminal to the remote machine named `hostname`.

Hostnames are listed in the `hosts` database, which may be contained in the /etc/hosts and /etc/inet/ipnodes files, the Network Information Service (NIS) hosts map, the Internet domain name server, or a combination of these. Each host has one official name (the first name in the database entry), and optionally one or more nicknames. Either official hostnames or nicknames may be specified in `hostname`.

Each remote machine may have a file named `/etc/hosts.equiv` containing a list of trusted hostnames with which it shares usernames. Users with the same username on both the local and remote machine may `rlogin` from the machines listed in the remote machine’s `/etc/hosts.equiv` file without supplying a password. Individual users may set up a similar private equivalence list with the file `.rhosts` in their home directories. Each line in this file contains two names: a `hostname` and a `username` separated by a space. An entry in a remote user’s `.rhosts` file permits the user named `username` who is logged into `hostname` to log in to the remote machine as the remote user without supplying a password. If the name of the local host is not found in the `/etc/hosts.equiv` file on the remote machine, and the local username and hostname are not found in the remote user’s `.rhosts` file, then the remote machine will prompt for a password. Hostnames listed in `/etc/hosts.equiv` and `.rhosts` files must be the official hostnames listed in the hosts database; nicknames may not be used in either of these files.

For security reasons, the `.rhosts` file must be owned by either the remote user or by root.

The remote terminal type is the same as your local terminal type (as given in your environment `TERM` variable). The terminal or window size is also copied to the remote system if the server supports the option, and changes in size are reflected as well. All echoing takes place at the remote site, so that (except for delays) the remote login is transparent. Flow control using CTRL-S and CTRL-Q and flushing of input and output on interrupts are handled properly.

**OPTIONS**  
The following options are supported:

- `-8`  
  Pass eight-bit data across the net instead of seven-bit data.

- `-ec`  
  Specify a different escape character, `c`, for the line used to disconnect from the remote host.

- `-E`  
  Stop any character from being recognized as an escape character.

- `-l name`  
  Specify a different `username` for the remote login. If you do not use this option, the remote username used is the same as your local username.

- `-L`  
  Allow the `rlogin` session to be run in “litout” mode.
**rlogin(1)**

### Escape Sequences

Lines that you type which start with the tilde character are “escape sequences” (the escape character can be changed using the `-e` option):

- `~.` Disconnect from the remote host. This is not the same as a logout, because the local host breaks the connection with no warning to the remote end.

- `-susp` Suspend the login session (only if you are using a shell with Job Control). `susp` is your “suspend” character, usually CTRL-Z; see `tty(1)`.

- `-dsusp` Suspend the input half of the login, but output will still be seen (only if you are using a shell with Job Control). `dsusp` is your “deferred suspend” character, usually CTRL-Y; see `tty(1)`.

### OPERANDS

`hostname` The remote machine on which `rlogin` establishes the remote login session.

### FILES

- `/etc/passwd` contains information about users’ accounts
- `/usr/hosts/*` for `hostname` version of the command
- `/etc/hosts.equiv` list of trusted hostnames with shared usernames
- `/etc/nologin` message displayed to users attempting to login during machine shutdown
- `$HOME/.rhosts` private list of trusted hostname/username combinations
- `/etc/hosts` hosts database
- `/etc/inet/ipnodes` hosts database

### ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWrcmdc</td>
</tr>
</tbody>
</table>

### SEE ALSO

- `rsh(1)`, `stty(1)`, `tty(1)`, `in.named(1M)`, `hosts(4)`, `nodes(4)`, `hosts.equiv(4)`, `nologin(4)`, `attributes(5)`

### DIAGNOSTICS

The following message indicates that the machine is in the process of being shutdown and logins have been disabled:

```
NO LOGINS: System going down in N minutes
```

### NOTES

When a system is listed in `hosts.equiv`, its security must be as good as local security. One insecure system listed in `hosts.equiv` can compromise the security of the entire system.
The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

This implementation can only use the TCP network service.
### NAME
rm, rmdir – remove directory entries

### SYNOPSIS
- `/usr/bin/rm [-f] [-i] file...
- `/usr/bin/rm -rR [-f] [-i] dirname... [file...]
- `/usr/xpg4/bin/rm [-fiRr] file...
- `/usr/bin/rmdir [-ps] dirname...

### /usr/bin/rm
The `rm` utility removes the directory entry specified by each `file` argument. If a file has no write permission and the standard input is a terminal, the full set of permissions (in octal) for the file are printed followed by a question mark. This is a prompt for confirmation. If the answer begins with `y` (for yes), the file is deleted, otherwise the file remains.

If `file` is a symbolic link, the link will be removed, but the file or directory to which it refers will not be deleted. Users do not need write permission to remove a symbolic link, provided they have write permissions in the directory.

If multiple `files` are specified and removal of a `file` fails for any reason, `rm` will write a diagnostic message to standard error, do nothing more to the current `file`, and go on to any remaining `files`.

If the standard input is not a terminal, the utility will operate as if the `-f` option is in effect.

### /usr/bin/rmdir
The `rmdir` utility will remove the directory entry specified by each `dirname` operand, which must refer to an empty directory.

Directories will be processed in the order specified. If a directory and a subdirectory of that directory are specified in a single invocation of `rmdir`, the subdirectory must be specified before the parent directory so that the parent directory will be empty when `rmdir` tries to remove it.

### OPTIONS
The following options are supported for `/usr/bin/rm` and `/usr/xpg4/bin/rm`:

- `-r` Recursively removes directories and subdirectories in the argument list. The directory will be emptied of files and removed. The user is normally prompted for removal of any write-protected files which the directory contains. The write-protected files are removed without prompting, however, if the `-f` option is used, or if the standard input is not a terminal and the `-i` option is not used.

  Symbolic links that are encountered with this option will not be traversed.

  If the removal of a non-empty, write-protected directory is attempted, the utility will always fail (even if the `-f` option is used), resulting in an error message.

- `-R` Same as `-r` option.
The following options are supported for /usr/bin/rm only:

- **-f**  Removes all files (whether write-protected or not) in a directory without prompting the user. In a write-protected directory, however, files are never removed (whatever their permissions are), but no messages are displayed. If the removal of a write-protected directory is attempted, this option will not suppress an error message.

- **-i**  Interactive. With this option, rm prompts for confirmation before removing any files. It overrides the -f option and remains in effect even if the standard input is not a terminal.

The following options are supported for /usr/xpg4/bin/rm only:

- **-f**  Does not prompt for confirmation. Does not write diagnostic messages or modify the exit status in the case of non-existent operands. Any previous occurrences of the -i option will be ignored.

- **-i**  Prompts for confirmation. Any occurrences of the -f option will be ignored.

The following options are supported for /usr/bin/rmdir only:

- **-p**  Allows users to remove the directory dirname and its parent directories which become empty. A message is printed to standard error if all or part of the path could not be removed.

- **-s**  Suppresses the message printed on the standard error when -p is in effect.

**OPERANDS**

The following operands are supported:

- **file**  A path name of a directory entry to be removed.

- **dirname**  A path name of an empty directory to be removed.

**USAGE**

See largefile(5) for the description of the behavior of rm and rmdir when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

**EXAMPLE 1** Removing directories

The following command:

```bash
example% rm a.out core
```

removes the directory entries a.out and core.

**EXAMPLE 2** Removing a directory without prompting

The following command:

```bash
example% rm -rf junk
```
EXAMPLE 2 Removing a directory without prompting  

removes the directory junk and all its contents, without prompting.

EXAMPLE 3 Removing empty directories

If a directory a in the current directory is empty, except that it contains a directory b, and a/b is empty except that it contains a directory c,

```
example% rmdir -p a/b/c
```

will remove all three directories.

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of rm and rmdir: LC_COLLATE, LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS

The following exit values are returned:

0      If the -f option was not specified, all the named directory entries were removed; otherwise, all the existing named directory entries were removed.

>0     An error occurred.

ATTRIBUTES

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

```
<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWxcu4</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>
```

SEE ALSO

rmdir(2), unlink(2), attributes(5), environ(5), largefile(5), XPG4(5)

DIAGNOSTICS

All messages are generally self-explanatory.

It is forbidden to remove the files "." and ". ." in order to avoid the consequences of inadvertently doing something like the following:

```
example% rm -r .*
```
NOTES
A – permits the user to mark explicitly the end of any command line options, allowing
rm to recognize file arguments that begin with a –. As an aid to BSD migration, rm will
accept – – as a synonym for –. This migration aid may disappear in a future release. If
a – – and a – both appear on the same command line, the second will be interpreted
as a file.
rmformat(1)

NAME
rmformat – removable rewritable media format utility

SYNOPSIS
rmformat [-DeHpUv] [-b label] [-c blockno] [-Fquick | long | force ]
 [-R enable | disable ] [-s filename] [-w enable | disable ] [-W enable | disable ]
 [devname]
rmformat -V read | write devname

DESCRIPTION
The rmformat utility is used to format, label, partition, and perform other
miscellaneous functions on removable, rewritable media that include
floppy drives, IOMEGA Zip/Jaz products, and the PCMCIA memory and ata cards. This utility can
also be used for the verification and surface analysis and for repair of the bad sectors
found during verification if the drive or the driver supports bad block management.

rmformat provides functionality to read/write protect the media with or without a
password. The password protection enabling or disabling is possible only with
selective rewritable media such as the IOMEGA Zip/Jaz products.

After formatting, rmformat writes the label, which covers the full capacity of the
media as one slice on floppy and PCMCIA memory cards to maintain compatibility
with the behavior of fdformat. On Zip/Jaz devices, the driver exports one slice
covering the full capacity of the disk as default. rmformat does not write the label on
Zip/Jaz media, unless explicitly requested. The partition information can be changed
with the help of other options provided by rmformat.

OPTIONS
The following options are supported:

- b label
  Labels the media with a SUNOS label. A SUNOS
  volume label name is restricted to 8 characters. For
  writing a DOS Volume label, the user should use
  mkfs_pcfs(1M).

- c blockno
  Corrects and repairs the given block. This correct and
  repair option may not be applicable to all devices
  supported by rmformat, as some devices may have a
  drive with bad block management capability and
  others may have this option implemented in the driver.
  If the drive or driver supports bad block management,
  a best effort is made to rectify the bad block. If the bad
  block still cannot be rectified, a message is displayed to
  indicate the failure to repair. The block number can be
  provided in decimal, octal, or hexadecimal format.

  The normal floppy and PCMCIA memory and ata
  cards do not support bad block management.

- D
  Formats a 720KB (3.5 inch) double density diskette.
  This is the default for double density type drives. This
  option is needed if the drive is a high or
  extended-density type.
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>-e</strong></td>
<td>Ejects the media upon completion. This feature may not be available if the drive does not support motorized eject.</td>
</tr>
<tr>
<td>**-F quick</td>
<td>long</td>
</tr>
<tr>
<td><strong>-H</strong></td>
<td>Formats a 1.44 MB (3.5 inch) high density diskette. This is the default for high density type drives. It is needed if the drive is the Extended Density type.</td>
</tr>
<tr>
<td><strong>-p</strong></td>
<td>Prints the protection status of the media. This option prints information whether the media is write, read, or password protected.</td>
</tr>
<tr>
<td>**-R enable</td>
<td>disable**</td>
</tr>
</tbody>
</table>

In legacy media such as floppy drives, all options start a long format depending on the mode (Extended Density mode, High Density mode, or Double Density mode) with which the floppy drive operates by default. On PCMCIA memory cards, all options start a long format.

A password length of 32 bytes (maximum) is allowed for the IOMEGA products that support this feature. This option is applicable only for IOMEGA products. IOMEGA products do not allow read/write protection.
without a password. On the devices which do not have such software read/write protect facility, warnings indicating the non-availability of this feature are provided.

Enables the user to lay out the partition information in the SUNOS label.

The user should provide a file as input with information about each slice in a format providing byte offset, size required, tags, and flags, as follows:

\[ \text{slices: } n = \text{offset}, \text{size} \ [, \text{flags}, \text{tags}] \]

where \( n \) is the slice number, \( \text{offset} \) is the byte offset at which the slice \( n \) starts, and \( \text{size} \) is the required size for slice \( n \). Both \( \text{offset} \) and \( \text{size} \) must be a multiple of 512 bytes. These numbers can be represented as decimal, hexadecimal, or octal numbers. No floating point numbers are accepted. Details about maximum number of slices can be obtained from the System Administration Guide: Basic Administration.

To specify the \( \text{size} \) or \( \text{offset} \) in kilobytes, megabytes, or gigabytes, add \( \text{KB, MB, GB} \), respectively. A number without a suffix is assumed to be a byte offset. The flags are represented as follows:

\[ \text{wm = read-write, mountable} \]
\[ \text{wu = read-write, unmountable} \]
\[ \text{ru = read-only, unmountable} \]

The tags are represented as follows: unassigned, boot, root, swap, usr, backup, stand, var, home, alternates.

The tags and flags can be omitted from the four tuple when finer control on those values is not required. It is required to omit both or include both. If the tags and flags are omitted from the four tuple for a particular slice, a default value for each is assumed. The default value for flags is \( \text{wm} \) and for tags is \( \text{unassigned} \).

Either full tag names can be provided or an abbreviation for the tags can be used. The abbreviations can be the first two or more letters from the standard tag names. \text{rmformat} is case insensitive in handling the defined tags & flags.

Slice specifications are separated by :
For example:
slices: 0 = 0, 30MB, "wm", "home"
       1 = 30MB, 51MB
       2 = 0, 100MB, "wm", "backup"
       6 = 81MB, 19MB

rmformat does the necessary checking to detect any overlapping partitions or illegal requests to addresses beyond the capacity of the media under consideration. There can be only one slice information entry for each slice \( n \). If multiple slice information entries for the same slice \( n \) is provided, an appropriate error message is displayed. The slice 2 be the backup slice covering the whole disk capacity. The pound sign character, \#, can be used to describe a line of comments in the input file. If the line starts with \#, then rmformat ignores all the characters following \# until the end of the line.

Partitioning some of the media with very small capacity is permitted, but be cautious in using this option on such devices.

- \( \text{-U} \)
  Performs umount on any file systems and then formats. See mount(1M). This option unmounts all the mounted slices and issues a long format on the device requested.

- \( \text{-V} \text{ read | write} \)
  Verifies each block of media after format. The write verification is a destructive mechanism. The user is queried for confirmation before the verification is started. The output of this option is a list of block numbers, which are identified as bad.

  The read verification only verifies the blocks and report the blocks which are prone to errors.
  The list of block numbers displayed can be used with the \( \text{-c} \) option for repairing.

- \( \text{-w enable | disable} \)
  Enables or disables the write protection on media. On devices that do not have a software write protect facility, a message indicating non-availability of this feature is displayed.

- \( \text{-W enable | disable} \)
  Enables or disables write protection with password. This option always works in interactive mode, as a password is requested from the user to maintain security.
A maximum password length of 32 bytes is allowed for IOMEGA products that support this feature. On devices that do not have the write protection with password, the software displays appropriate messages indicating the non-availability of such features.

OPERANDS

The following operands are supported:

devname

devname can be provided as absolute device pathname or relative pathname for the device from the current working directory or the nickname as exported by the System Volume manager. See vold(1M).

For floppy device /dev/rdiskette0 (for system without volume management) or floppy0 (system with volume management) can be used. To use the first drive, specify /dev/rdiskette1 (for system without volume management), or floppy1 (system with volume management) to use the second drive.

For system without volume management running, user can also provide the absolute device pathname as /dev/rdisk/c?t?d? or the appropriate relative device pathname from the current working directory.

EXAMPLES

EXAMPLE 1 Formatting a diskette

```bash
example$ rmformat -F quick /dev/rdiskette

Formatting will erase all the data on disk.
Do you want to continue? (y/n)y
```

EXAMPLE 2 Formatting a Zip drive

```bash
example$ rmformat -F quick /vol/dev/aliases/zip0

Formatting will erase all the data on disk.
Do you want to continue? (y/n)y
```

EXAMPLE 3 Formatting a diskette for a UFS file system

The following example formats a diskette and creates a UFS file system:

```bash
example$ rmformat -F quick /vol/dev/aliases/floppy0

Formatting will erase all the data on disk.
Do you want to continue? (y/n)y
example$ su

# /usr/sbin/newfs /vol/dev/aliases/floppy0

newfs: construct a new file system /dev/rdiskette: (y/n)? y
/dev/rdiskette: 2880 sectors in 80 cylinders of 2 tracks, 18 sectors
1.4MB in 5 cyl groups (16 c/g, 0.28MB/g, 128 i/g)
super-block backups (for fsck -P ufs -o b=) at:
 32, 640, 1184, 1792, 2336,
```

#
EXAMPLE 4 Formatting removable media for a PCFS file system

The following example shows how to create an alternate fdisk partition:

```
example$ rmformat -F quick /dev/rdsk/c0t4d0s2:c
Formatting will erase all the data on disk.
Do you want to continue? (y/n)? y
example$ su
# fdisk /dev/rdsk/c0t4d0s2:c
# mkfs -F pcfs /dev/rdsk/c0t4d0s2:c
Construct a new FAT file system on /dev/rdsk/c0t4d0s2:c: (y/n)? y
#
```

The following example describes how to create a PCFS file system without an fdisk partition:

```
example$ rmformat -F quick /dev/rdiskette
Formatting will erase all the data on disk.
Do you want to continue? (y/n)? y
example$ su
# mkfs -F pcfs -o nofdisk,size=2 /dev/rdiskette
Construct a new FAT file system on /dev/rdiskette: (y/n)? y
#
```

EXAMPLE 5 Enabling or disabling read or write protection

The following example shows how to enable write protection and set a password on a Zip drive:

```
example$ rmformat -W enable /vol/dev/aliases/zip0
Please enter password (32 chars maximum): xxx
Please reenter password: xxx
```

The following example shows how to disable write protection and remove the password on a Zip drive:

```
example$ rmformat -W disable /vol/dev/aliases/zip0
Please enter password (32 chars maximum): xxx
```

The following example shows how to enable read protection and set a password on a Zip drive:

```
example$ rmformat -R enable /vol/dev/aliases/zip0
Please enter password (32 chars maximum): xxx
Please reenter password: xxx
```

The following example shows how to disable read protection and remove the password on a Zip drive:

```
example$ rmformat -R disable /vol/dev/aliases/zip0
Please enter password (32 chars maximum): xxx
```

FILES

```
/vol/dev/diskette0 Directory providing block device access for the media in floppy drive 0.
```
rmformat(1)

/vol/dev/rdiskette0 Directory providing character device access for the media in floppy drive 0.
/vol/dev/aliases Directory providing symbolic links to the character devices for the different media under the control of volume management using appropriate alias.
/vol/dev/aliases/floppy0 Symbolic link to the character device for the media in floppy drive 0.
/vol/dev/aliases/zip0 Symbolic link to the character device for the media in Zip drive 0.
/vol/dev/aliases/jaz0 Symbolic link to the character device for the media in Jaz drive 0.
/dev/rdiskette Symbolic link providing character device access for the media in the primary floppy drive, usually drive 0.
/vol/dev/dsk Directory providing block device access for the PCMCIA memory and ata cards and removable media devices.
/vol/dev/rdsk Directory providing character device access for the PCMCIA memory and ata cards and removable media devices.
/vol/dev/aliases/pcmemS Symbolic link to the character device for the PCMCIA memory card in socket S, where S represents a PCMCIA socket number.
/vol/dev/aliases/rmdisk0 Symbolic link to the generic removable media device that is not a Zip, Jaz, CD-ROM, floppy, DVD-ROM, PCMCIA memory card, and so forth.
/dev/rdsk Directory providing character device access for the PCMCIA memory and ata cards and other removable devices.
/dev/dsk Directory providing block device access for the PCMCIA memory and ata cards and other removable media devices.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>
A rewritable media or PCMCIA memory card or PCMCIA ata card containing a ufs file system created on a SPARC-based system (using newfs(1M)) is not identical to a rewritable media or PCMCIA memory card containing a ufs file system created on an IA based system. Do not interchange any removable media containing ufs between these platforms; use cpio(1) or tar(1) to transfer files on diskettes or memory cards between them. For interchangeable filesystems refer to pcfs(7FS) and udfs(7FS).

Currently, bad sector mapping is not supported on floppy diskettes or PCMCIA memory cards. Therefore, a diskette or memory card is unusable if rmformat finds an error (bad sector).
NAME  roffbib – format and print a bibliographic database


DESCRIPTION  roffbib prints out all records in a bibliographic database, in bibliography format rather than as footnotes or endnotes. Generally it is used in conjunction with sortbib(1):

example% sortbib database | roffbib

OPTIONS  roffbib accepts all options understood by nroff(1) except -i and -q.

  -e   Produce equally-spaced words in adjusted lines using full terminal resolution.

  -h   Use output tabs during horizontal spacing to speed output and reduce output character count. TAB settings are assumed to be every 8 nominal character widths.

  -m filename  Prepend the macro file /usr/share/lib/tmac/tmac.name to the input files. There should be a space between the -m and the macro filename. This set of macros will replace the ones defined in /usr/share/lib/tmac/tmac.bib.

  -np   Number first generated page p.

  -olist  Print only page numbers that appear in the comma-separated list of numbers and ranges. A range N–M means pages N through M; an initial -N means from the beginning to page N; a final N– means from page N to end.

  -Q  Queue output for the phototypesetter. Page offset is set to 1 inch.

  -raN  Set register a (one-character) to N. The command-line argument -raN1 will number the references starting at 1.

Four command-line registers control formatting style of the bibliography, much like the number registers of ms(5). The flag -rV2 will double space the bibliography, while -rV1 will double space references but single space annotation paragraphs. The line length can be changed from the default 6.5 inches to 6 inches with the -rL6i argument, and the page offset can be set from the default of 0 to one inch by specifying -rO1i (capital O, not zero).

  -sN  Halt prior to every N pages for paper loading or changing (default N =1). To resume, enter NEWLINE or RETURN.

  -Tterm  Specify term as the terminal type.

  -V  Send output to the Versatec. Page offset is set to 1 inch.
If abstracts or comments are entered following the %x field key, roffbib will format them into paragraphs for an annotated bibliography. Several %x fields may be given if several annotation paragraphs are desired.

FILES
/usr/share/lib/tmac/tmac.bib  file of macros used by nroff/troff

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWdoc</td>
</tr>
</tbody>
</table>

SEE ALSO
addbib(1), indxbib(1), lookbib(1), nroff(1) refer(1), sortbib(1), troff(1), attributes(5)

BUGS
Users have to rewrite macros to create customized formats.

roffbib(1)
The command `roles` prints on standard output the roles that you or the optionally-specified user have been granted. Roles are special accounts that correspond to a functional responsibility rather than to an actual person (referred to as a normal user).

Each user may have zero or more roles. Roles have most of the attributes of normal users and are identified like normal users in `passwd(4)` and `shadow(4)`. Each role must have an entry in the `user_attr(4)` file that identifies it as a role. Roles can have their own authorizations and profiles. See `auths(1)` and `profiles(1)`.

Roles are not allowed to log into a system as a primary user. Instead, a user must log in as him—or herself and assume the role. The actions of a role are attributable to the normal user. When auditing is enabled, the audited events of the role contain the audit ID of the original user who assumed the role.

A role may not assume itself or any other role. Roles are not hierarchical. However, rights profiles (see `prof_attr(4)`) are hierarchical and can be used to achieve the same effect as hierarchical roles.

Roles must have valid passwords and one of the shells that interprets profiles: either `pfcsh`, `pfksh`, or `pfsh`. See `pfexec(1)`.

Role assumption may be performed using `su(1M)`, `rlogin(1)`, or some other service that supports the `PAM_RUSER` variable. Successful assumption requires knowledge of the role’s password and membership in the role. Role assignments are specified in `user_attr(4)`.

**EXAMPLE 1** Sample output

The output of the `roles` command has the following form:

```
example% roles tester01 tester02
 tester01 : admin
 tester02 : secadmin, root
example%
```

**EXIT STATUS**

The following exit values are returned:

- 0  Successful completion.
- 1  An error occurred.

**FILES**

- `/etc/user_attr`
- `/etc/security/auth_attr`
- `/etc/security/prof_attr`

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

**SEE ALSO**

auths(1), pfexec(1), profiles(1), rlogin(1), su(1M), getauusername(3BSM), auth_attr(4), passwd(4), prof_attr(4), shadow(4), user_attr(4), attributes(5)
rpcgen(1)

NAME
rpcgen – an RPC protocol compiler

SYNOPSIS
rpcgen infile


rpcgen [-c | -h | -l | -m | -t | -Sc | -Ss | -Sm] [-o outfile] [infile]

rpcgen [-s nettype] [-o outfile] [infile]

rpcgen [-n netid] [-o outfile] [infile]

DESCRIPTION
The rpcgen utility is a tool that generates C code to implement an RPC protocol. The input to rpcgen is a language similar to C known as RPC Language (Remote Procedure Call Language).

The rpcgen utility is normally used as in the first synopsis where it takes an input file and generates three output files. If the infile is named proto.x, then rpcgen generates a header in proto.h, XDR routines in proto_xdr.c, server-side stubs in proto_svc.c, and client-side stubs in proto_clnt.c. With the -T option, it also generates the RPC dispatch table in proto_tbl.i.

rpcgen can also generate sample client and server files that can be customized to suit a particular application. The -Sc, -Ss, and -Sm options generate sample client, server and makefile, respectively. The -a option generates all files, including sample files. If the infile is proto.x, then the client side sample file is written to proto_client.c, the server side sample file to proto_server.c and the sample makefile to makefile.proto.

The server created can be started both by the port monitors (for example, inetd or listen) or by itself. When it is started by a port monitor, it creates servers only for the transport for which the file descriptor 0 was passed. The name of the transport must be specified by setting up the environment variable PM_TRANSPORT. When the server generated by rpcgen is executed, it creates server handles for all the transports specified in the NETPATH environment variable, or if it is unset, it creates server handles for all the visible transports from the /etc/netconfig file. Note: the transports are chosen at run time and not at compile time. When the server is self-started, it backgrounds itself by default. A special define symbol RPC_SVC_FG can be used to run the server process in foreground.

The second synopsis provides special features which allow for the creation of more sophisticated RPC servers. These features include support for user-provided #defines and RPC dispatch tables. The entries in the RPC dispatch table contain:

- pointers to the service routine corresponding to that procedure
- a pointer to the input and output arguments
- the size of these routines

A server can use the dispatch table to check authorization and then to execute the service routine. A client library may use the dispatch table to deal with the details of storage management and XDR data conversion.
The other three synopses shown above are used when one does not want to generate all the output files, but only a particular one. See the EXAMPLES section below for examples of rpcgen usage. When rpcgen is executed with the -s option, it creates servers for that particular class of transports. When executed with the -n option, it creates a server for the transport specified by netid. If infile is not specified, rpcgen accepts the standard input.

All the options mentioned in the second synopsis can be used with the other three synopses, but the changes will be made only to the specified output file.

The C preprocessor cc -E is run on the input file before it is actually interpreted by rpcgen. For each type of output file, rpcgen defines a special preprocessor symbol for use by the rpcgen programmer:

- RPC_HDR
  - defined when compiling into headers
- RPC_XDR
  - defined when compiling into XDR routines
- RPC_SVC
  - defined when compiling into server-side stubs
- RPC_CLNT
  - defined when compiling into client-side stubs
- RPC_TBL
  - defined when compiling into RPC dispatch tables

Any line beginning with "%%" is passed directly into the output file, uninterpreted by rpcgen, except that the leading "%%" is stripped off. To specify the path name of the C preprocessor, use the -Y flag.

For every data type referred to in infile, rpcgen assumes that there exists a routine with the string xdr_ prepended to the name of the data type. If this routine does not exist in the RPC/XDR library, it must be provided. Providing an undefined data type allows customization of XDR routines.

**OPTIONS**

The following options are supported:

- `-a`
  - Generates all files, including sample files.
- `-A`
  - Enables the Automatic MT mode in the server main program. In this mode, the RPC library automatically creates threads to service client requests. This option generates multithread-safe stubs by implicitly turning on the -M option. Server multithreading modes and parameters can be set using the rpc_control(3NSL) call. rpcgen generated code does not change the default values for the Automatic MT mode.
- `-b`
  - Backward compatibility mode. Generates transport-specific RPC code for older versions of the operating system.
- `-c`
  - Compiles into XDR routines.
- `-C`
  - Generates header and stub files which can be used with ANSI C compilers. Headers generated with this flag can also be used with C++ programs.
rpcgen(1)

-Dname [=value]  Defines a symbol name. Equivalent to the \#define directive in the source. If no value is given, value is defined as 1. This option may be specified more than once.

-h  Compiles into C data-definitions (a header). The -T option can be used in conjunction to produce a header which supports RPC dispatch tables.

-i size  Size at which to start generating inline code. This option is useful for optimization. The default size is 5.

-I  Compiles support for inetd(1M) in the server side stubs. Such servers can be self-started or can be started by inetd. When the server is self-started, it backgrounds itself by default. A special define symbol RPC_SVC_FG can be used to run the server process in foreground, or the user may simply compile without the -I option.

If there are no pending client requests, the inetd servers exit after 120 seconds (default). The default can be changed with the -K option. All of the error messages for inetd servers are always logged with syslog(3C).

Note: This option is supported for backward compatibility only. It should always be used in conjunction with the -b option which generates backward compatibility code. By default (that is, when -b is not specified), rpcgen generates servers that can be invoked through portmonitors.

-K seconds  By default, services created using rpcgen and invoked through port monitors wait 120 seconds after servicing a request before exiting. That interval can be changed using the -K flag. To create a server that exits immediately upon servicing a request, use -K 0.

To create a server that never exits, the appropriate argument is -K -1.

When monitoring for a server, some portmonitors, like listen(1M), always spawn a new process in response to a service request. If it is known that a server will be used with such a monitor, the server should exit immediately on completion. For such servers, rpcgen should be used with -K 0.

-l  Compiles into client-side stubs.

-L  When the servers are started in foreground, uses syslog(3C) to log the server errors instead of printing them on the standard error.
-m Compiles into server-side stubs, but do not generate a "main" routine. This option is useful for doing callback-routines and for users who need to write their own "main" routine to do initialization.

-M Generates multithread-safe stubs for passing arguments and results between rpcgen-generated code and user written code. This option is useful for users who want to use threads in their code.

-N This option allows procedures to have multiple arguments. It also uses the style of parameter passing that closely resembles C. So, when passing an argument to a remote procedure, you do not have to pass a pointer to the argument, but can pass the argument itself. This behavior is different from the old style of rpcgen-generated code. To maintain backward compatibility, this option is not the default.

-n netid Compiles into server-side stubs for the transport specified by netid. There should be an entry for netid in the netconfig database. This option may be specified more than once, so as to compile a server that serves multiple transports.

-o outfile Specifies the name of the output file. If none is specified, standard output is used (-c, -h, -l, -m, -n, -s, -Sc, -Sm, -Ss, and -t modes only).

-s nettype Compiles into server-side stubs for all the transports belonging to the class nettype. The supported classes are netpath, visible, circuit_n, circuit_v, datagram_n, datagram_v, tcp, and udp (see rpc(3NSL) for the meanings associated with these classes). This option may be specified more than once. Note: The transports are chosen at run time and not at compile time.

-Sc Generates sample client code that uses remote procedure calls.

-Sm Generates a sample Makefile which can be used for compiling the application.

-Ss Generates sample server code that uses remote procedure calls.

-t Compiles into RPC dispatch table.

-T Generates the code to support RPC dispatch tables.

The options -c, -h, -l, -m, -n, -s, -Sc, -Sm, -Ss, and -t are used exclusively to generate a particular type of file, while the options -D and -T are global and can be used with the other options.

-v Displays the version number.

-Y pathname Gives the name of the directory where rpcgen will start looking for the C preprocessor.
The following operand is supported:

infile  input file

**EXAMPLES**

**EXAMPLE 1** Generating the output files and dispatch table

The following entry

```bash
example$ rpcgen -T prot.x
```

generates all the five files: `prot.h`, `prot_clnt.c`, `prot_svc.c`, `prot_xdr.c`, and `prot_tbl.i`.

**EXAMPLE 2** Sending headers to standard output

The following example sends the C data-definitions (header) to the standard output:

```bash
example$ rpcgen -h prot.x
```

**EXAMPLE 3** Sending a test version

To send the test version of the `DTEST`, server side stubs for all the transport belonging to the class `datagram_n` to standard output, use:

```bash
example$ rpcgen -s datagram_n -DTEST prot.x
```

**EXAMPLE 4** Creating server side stubs

To create the server side stubs for the transport indicated by `netid tcp`, use:

```bash
example$ rpcgen -n tcp -o prot_svc.c prot.x
```

**EXIT STATUS**

0  Successful operation.

>0  An error occurred.

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWbtool</td>
</tr>
</tbody>
</table>

**SEE ALSO**

`cc(1B), inetd(1M), listen(1M), rpc(3NSL), rpc_control(3NSL), rpc_svc_calls(3NSL), syslog(3C), netconfig(4), attributes(5)`

The `rpcgen` chapter in the *ONC+ Developer’s Guide* manual.
NAME
rpm2cpio – convert Red Hat Package (RPM) to cpio archive

SYNOPSIS
rpm2cpio [file.rpm]

DESCRIPTION
The rpm2cpio utility converts the .rpm file specified as its sole argument to a cpio archive on standard output. (See NOTES.) If no argument is given, an rpm stream is read from standard input. In both cases, rpm2cpio will fail and print a usage message if the standard output is a terminal. Therefore, the output is usually redirected to a file or piped through the cpio(1) utility.

EXAMPLES
EXAMPLE 1 Converting an rpm file
example® rpm2cpio Device3Dfx-1.1-2.src.rpm | cpio -itv
CPIO archive found!
-rw-r--r-- 1 root root 2635 Sep 13 16:39 1998, 3dfx.gif
-rw-r--r-- 1 root root 11339 Sep 27 16:03 1998, Dev3Dfx.tar.gz
-rw-r--r-- 1 root root 1387 Sep 27 16:04 1998, Device3Dfx-1.1-2.spec
31 blocks

EXAMPLE 2 Converting from standard input
example® rpm2cpio < Device3Dfx-1.1-2.src.rpm | cpio -itv
CPIO archive found!
-rw-r--r-- 1 root root 2635 Sep 13 16:39 1998, 3dfx.gif
-rw-r--r-- 1 root root 11339 Sep 27 16:03 1998, Dev3Dfx.tar.gz
-rw-r--r-- 1 root root 1387 Sep 27 16:04 1998, Device3Dfx-1.1-2.spec
31 blocks

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWrpm</td>
</tr>
</tbody>
</table>

SEE ALSO
rpm2cpio handles versions 3 and 4 RPMs.
**rsh** connects to the specified hostname and executes the specified command. **rsh** copies its standard input to the remote command, the standard output of the remote command to its standard output, and the standard error of the remote command to its standard error. Interrupt, quit, and terminate signals are propagated to the remote command. **rsh** normally terminates when the remote command does.

If you omit **command**, instead of executing a single command, **rsh** logs you in on the remote host using **rlogin**(1).

**rsh** will not return the exit status code of **command**.

Shell metacharacters which are not quoted are interpreted on the local machine, while quoted metacharacters are interpreted on the remote machine. See **EXAMPLES**.

If there is no locale setting in the initialization file of the login shell (**.cshrc**, ...) for a particular user, **rsh** always executes the command in the "C" locale instead of using the default locale of the remote machine.

**OPTIONS**

The following options are supported:

- **-l username**
  Uses **username** as the remote username instead of your local username. In the absence of this option, the remote username is the same as your local username.

- **-n**
  Redirects the input of **rsh** to **/dev/null**. You sometimes need this option to avoid unfortunate interactions between **rsh** and the shell which invokes it. For example, if you are running **rsh** and invoke a **rsh** in the background without redirecting its input away from the terminal, it will block even if no reads are posted by the remote command. The **-n** option will prevent this.

The type of remote shell (**sh**, **rsh**, or other) is determined by the user’s entry in the file **/etc/passwd** on the remote system.

**OPERANDS**

The following operand is supported:

- **command**
  The command to be executed on the specified hostname.

**USAGE**

See **largefile**(5) for the description of the behavior of **rsh** and **remsh** when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).
The `rsh` and `remsh` commands are IPv6-enabled. See `ip6(7P)`.

Hostnames are given in the `hosts` database, which may be contained in the `/etc/hosts` file, the Internet domain name database, or both. Each host has one official name (the first name in the database entry) and optionally one or more nicknames. Official hostnames or nicknames may be given as `hostname`.

If the name of the file from which `rsh` is executed is anything other than `rsh`, `rsh` takes this name as its `hostname` argument. This allows you to create a symbolic link to `rsh` in the name of a host which, when executed, will invoke a remote shell on that host. By creating a directory and populating it with symbolic links in the names of commonly used hosts, then including the directory in your shell’s search path, you can run `rsh` by typing `hostname` to your shell.

If `rsh` is invoked with the basename `remsh`, `rsh` will check for the existence of the file `/usr/bin/remsh`. If this file exists, `rsh` will behave as if `remsh` is an alias for `rsh`. If `/usr/bin/remsh` does not exist, `rsh` will behave as if `remsh` is a host name.

Each remote machine may have a file named `/etc/hosts.equiv` containing a list of trusted hostnames with which it shares usernames. Users with the same username on both the local and remote machine may run `rsh` from the machines listed in the remote machine’s `/etc/hosts` file. Individual users may set up a similar private equivalence list with the file `.rhosts` in their home directories. Each line in this file contains two names: a `hostname` and a `username` separated by a space. The entry permits the user named `username` who is logged into `hostname` to use `rsh` to access the remote machine as the remote user. If the name of the local host is not found in the `/etc/hosts.equiv` file on the remote machine, and the local username and hostname are not found in the remote user’s `.rhosts` file, then the access is denied. The hostnames listed in the `/etc/hosts.equiv` and `.rhosts` files must be the official hostnames listed in the `hosts` database; nicknames may not be used in either of these files.

`rsh` will not prompt for a password if access is denied on the remote machine unless the `command` argument is omitted.

### EXAMPLES

**EXAMPLE 1** Using `rsh` to append files

The following command:

```bash
example% rsh lizard cat lizard.file >> example.file
```

appends the remote file `lizard.file` from the machine called `lizard` to the file called `example.file` on the machine called `example`, while the command:

```bash
example% rsh lizard cat lizard.file """"" >>"" lizard.file2
```

appends the file `lizard.file` on the machine called `lizard` to the file `lizard.file2` which also resides on the machine called `lizard`.  

User Commands 1279
rsh(1)

**EXIT STATUS** The following exit values are returned:
0 Successful completion.
1 An error occurred.

**FILES**
- `/etc/hosts` Internet host table
- `/etc/hosts.equiv` trusted remote hosts and users
- `/etc/passwd` system password file

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWrcmdc</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

**SEE ALSO**
- `on(1)`, `rlogin(1)`, `telnet(1)`, `vi(1)`, `in.named(1M)`, `hosts(4)`, `hosts.equiv(4)`, `ipnodes(4)`, `attributes(5)`, `largefile(5)`, `ip6(7P)`

**NOTES**
When a system is listed in `hosts.equiv`, its security must be as good as local security. One insecure system listed in `hosts.equiv` can compromise the security of the entire system.

You cannot run an interactive command (such as `vi(1)`). Use `rlogin` if you wish to do this.

Stop signals stop the local `rsh` process only. This is arguably wrong, but currently hard to fix for reasons too complicated to explain here.

The current local environment is not passed to the remote shell.

Sometimes the `-n` option is needed for reasons that are less than obvious. For example, the command:

```bash
example$ rsh somehost dd if=/dev/nrmt0 bs=20b | tar xvpsf -n will put your shell into a strange state. Evidently, what happens is that the `tar` terminates before the `rsh`. The `rsh` then tries to write into the “broken pipe” and, instead of terminating neatly, proceeds to compete with your shell for its standard input. Invoking `rsh` with the `-n` option avoids such incidents.
```

This bug occurs only when `rsh` is at the beginning of a pipeline and is not reading standard input. Do not use the `-n` if `rsh` actually needs to read standard input. For example,

```bash
example$ tar cf - . | rsh sundial dd of=/dev/rmt0 bs=20b does not produce the bug. If you were to use the `-n` in a case like this, `rsh` would incorrectly read from `/dev/null` instead of from the pipe.
```
NAME
run – run an executable

SYNOPSIS
run [-s] [-e] [-n] [-t string] program

DESCRIPTION
The run command runs program, using the PATH variable to find it. By default, when program has completed, the user is prompted (Press ENTER to continue), before being returned to FMLI. The argument program is a system executable followed by its options (if any).

OPTIONS
The following options are supported:

-e
If -e is specified, the user will be prompted before returning to FMLI only if there is an error condition.

-n
If -n is specified, the user will never be prompted before returning to FMLI (useful for programs like vi, in which the user must do some specific action to exit in the first place).

-s
The -s option means "silent", implying that the screen will not have to be repainted when program has completed. Note that the -s option should only be used when program does not write to the terminal. In addition, when -s is used, program cannot be interrupted, even if it recognizes interrupts.

-t string
If -t is specified, string is the name this process will have in the pop-up menu generated by the frm-list command. This feature requires the executable facesuspend (see face(1)) to suspend the process and return to the FMLI application.

EXAMPLES

EXAMPLE 1 Sample output of run command

Here is a menu that uses run:

menu="Edit special System files"
name="Password file"
action=’run -e vi /etc/passwd’
name="Group file"
action=’run -e vi /etc/group’
name="My .profile"
action=’run -n vi $HOME/.profile’

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>
run(1F)

SEE ALSO  face(1), attributes(5)
runat(1)

NAME
runat – execute command in extended attribute name space

SYNOPSIS
/usr/bin/runat file [command]

DESCRIPTION
The runat utility is used to execute shell commands in a file’s hidden attribute
directory. Effectively, this utility changes the current working directory to be the
hidden attribute directory associated with the file argument and then executes the
specified command in the bourne shell (/bin/sh). If no command argument is
provided, an interactive shell is spawned. The environment variable $SHELL defines
the shell to be spawned. If this variable is undefined, the default shell, /bin/sh, is
used.

The file argument can be any file, including a directory, that can support extended
attributes. It is not necessary that this file have any attributes (or be prepared in any
way) before invoking the runat command.

OPERANDS
The following operands are supported:

file Any file, including a directory, that can support extended
         attributes.
command The command to be executed in an attribute directory.

ERRORS
A non-zero exit status will be returned if runat cannot access the file argument, or the
file argument does not support extended attributes.

USAGE
See fsattr(5) for a detailed description of extended file attributes.

The process context created by the runat command has its current working directory
set to the hidden directory containing the file’s extended attributes. The parent of this
directory (the "..") entry) always refers to the file provided on the command line. As
such, it may not be a directory. Therefore, commands (such as pwd) that depend upon
the parent entry being well-formed (that is, referring to a directory) may fail.

In the absence of the command argument, runat will spawn a new interactive shell
with its current working directory set to be the provided file’s hidden attribute
directory. Notice that some shells (such as zsh and tcsh) are not well behaved when
the directory parent is not a directory, as described above. These shells should not be
used with runat.

EXAMPLES
EXAMPLE 1 Using runat to list extended attributes on a file
example1 runat file.1 ls -l
example1 runat file.1 ls

EXAMPLE 2 Creating extended attributes
example2 runat file.2 cp /tmp/attrdata attr.1
example2 runat file.2 cat /tmp/attrdata > attr.1
EXAMPLE 3 Copying an attribute from one file to another

```bash
example$ runat file.2 cat attr.1 | runat file.1 "cat > attr.1"
```

EXAMPLE 4 Using runat to spawn an interactive shell

```bash
example$ runat file.3 /bin/sh
```

This spawns a new shell in the attribute directory for `file.3`. Notice that the shell will not be able to determine what your current directory is. To leave the attribute directory, either exit the spawned shell or change directory (`cd`) using an absolute path.

Recommended methods for performing basic attribute operations:

- **display**
  - `runat file ls [options]`

- **read**
  - `runat file cat attribute`

- **create/modify**
  - `runat file cp absolute-file-path attribute`

- **delete**
  - `runat file rm attribute`

- **permission changes**
  - `runat file chmod mode attribute`
  - `runat file chgrp group attribute`
  - `runat file chown owner attribute`

- **interactive shell**
  - `runat file /bin/sh`
  - or set your `$SHELL` to `/bin/sh` and `runat file`

The above list of commands are known to work with `runat`. While many other commands may work, there is no guarantee that any beyond this list will work. Any command that relies on being able to determine its current working directory is likely to fail. Examples of such commands follow:

EXAMPLE 5 Using man in an attribute directory

```bash
example$ runat file.1 man runat
getcwd: Not a directory
```

EXAMPLE 6 Spawning a tcsh shell in an attribute directory

```bash
example$ runat file.3 /usr/bin/tcsh
tcsh: Not a directory
tcsh: Trying to start from "/home/user"
```
EXAMPLE 6 Spawning a tcsh shell in an attribute directory (Continued)

A new tcsh shell has been spawned with the current working directory set to the user’s home directory.

EXAMPLE 7 Spawning a zsh shell in an attribute directory

element runat file.3 /usr/bin/zsh
element

While the command appears to have worked, zsh has actually just changed the current working directory to ‘/’. This can be seen by using /bin/pwd:

element /bin/pwd

ENVIRONMENT VARIABLES

SHELL Specifies the command shell to be invoked by runat.

EXIT STATUS

The following exit values are returned:

125 The attribute directory of the file referenced by the file argument cannot be accessed.

126 The exec of the provided command argument failed.

Otherwise, the exit status returned is the exit status of the shell invoked to execute the provided command.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Evolving</td>
</tr>
</tbody>
</table>

SEE ALSO

open(2), attributes(5), fsattr(5)

NOTES

It is not always obvious why a command fails in runat when it is unable to determine the current working directory. The errors resulting can be confusing and ambiguous (see the tcsh and zsh examples above).
rup(1)

NAME    rup – show host status of remote machines (RPC version)
SYNOPSIS rup [-hlt]
            rup [host...]
DESCRIPTION rup gives a status similar to uptime for remote machines. It broadcasts on the local
network, and displays the responses it receives.

Normally, the listing is in the order that responses are received, but this order can be
changed by specifying one of the options listed below.

When host arguments are given, rather than broadcasting rup will only query the list
of specified hosts.

A remote host will only respond if it is running the rstatd daemon, which is
normally started up from inetd(1M).

OPTIONS -h    Sort the display alphabetically by host name.
           -l    Sort the display by load average.
           -t    Sort the display by up time.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWrcmdc</td>
</tr>
</tbody>
</table>

SEE ALSO ruptime(1), inetd(1M), attributes(5)

Solaris 9 Installation Guide

BUGS Broadcasting does not work through gateways.
NAME
rup – show host status of remote machines (RPC version)

SYNOPSIS
rup [-hlt]

rup [host...]

DESCRIPTION
rup gives a status similar to uptime for remote machines. It broadcasts on the local network, and displays the responses it receives.

Normally, the listing is in the order that responses are received, but this order can be changed by specifying one of the options listed below.

When host arguments are given, rather than broadcasting rup only queries the list of specified hosts.

A remote host will only respond if it is running the rstatd daemon, which is normally started up from inetd(1M).

OPTIONS
-h Sort the display alphabetically by host name.
-l Sort the display by load average.
-t Sort the display by up time.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
</tbody>
</table>

SEE ALSO
rup(1C), runtime(1), inetd(1M), attributes(5)

BUGS
Broadcasting does not work through gateways.
ruptime(1)

NAME  
ruptime – show host status of local machines

SYNOPSIS  
ruptime [-ar] [-l | -t | -u]

DESCRIPTION  
The ruptime utility gives a status line like uptime (see uptime(1)) for each machine on the local network; these are formed from packets broadcast by each host on the network approximately every three minutes.

Machines for which no status report has been received for 11 minutes are shown as being down.

Normally, the listing is sorted by host name, but this order can be changed by specifying one of the options listed below.

OPTIONS  
The following options are supported:
-  
a
  Counts even those users who have been idle for an hour or more.
  
r
  Reverses the sorting order.
  
-  
  l | -t | -u
  These options are mutually exclusive. The use of one overrides the previous one(s).
    
    - l
      Sorts the display by load average.
    
    - t
      Sorts the display by up time.
    
    - u
      Sorts the display by number of users.

FILES  
/var/spool/rwho/whod.* data files

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWrcmde</td>
</tr>
</tbody>
</table>

SEE ALSO  
uptime(1), rwho(1), in.rwhod(1M), attributes(5)
NAME | rusage – print resource usage for a command
SYNOPSIS | /usr/ucb/rusage command
DESCRIPTION | The rusage command is similar to time(1). It runs the given command, which must be specified; that is, command is not optional as it is in the C shell’s timing facility. When the command is complete, rusage displays the real (wall clock), the system CPU, and the user CPU times which elapsed during execution of the command, plus other fields in the rusage structure, all on one long line. Times are reported in seconds and hundredths of a second.

EXAMPLES | **EXAMPLE 1** The format of rusage output

The example below shows the format of rusage output.
```
example% rusage wc /usr/share/man/man1/csh (1)
3045 13423 78071 /usr/share/man/man1/csh (1)
2.26 real 0.80 user 0.36 sys 11 pf 38 pr 0 sw 11 rb 0 wb 16 vcx 37
  icx 24 mx 0 ix 1230 id 9 is
example%
```

Each of the fields identified corresponds to an element of the rusage structure, as described in getrusage(3C), as follows:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>real</td>
<td>elapsed real time</td>
</tr>
<tr>
<td>user</td>
<td>user time used</td>
</tr>
<tr>
<td>sys</td>
<td>system time used</td>
</tr>
<tr>
<td>pf</td>
<td>page faults requiring physical I/O</td>
</tr>
<tr>
<td>pr</td>
<td>page faults not requiring physical I/O</td>
</tr>
<tr>
<td>sw</td>
<td>swaps</td>
</tr>
<tr>
<td>rb</td>
<td>block input operations</td>
</tr>
<tr>
<td>wb</td>
<td>block output operations</td>
</tr>
<tr>
<td>vcx</td>
<td>voluntary context switches</td>
</tr>
<tr>
<td>icx</td>
<td>involuntary context switches</td>
</tr>
<tr>
<td>mx</td>
<td>maximum resident set size</td>
</tr>
<tr>
<td>ix</td>
<td>currently 0</td>
</tr>
<tr>
<td>id</td>
<td>integral resident set size</td>
</tr>
<tr>
<td>is</td>
<td>currently 0</td>
</tr>
</tbody>
</table>

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:
When the command being timed is interrupted, the timing values displayed may be inaccurate.

SEE ALSO
csh(1), time(1), getrusage(3C), attributes(5)

BUGS
rusers(1)

NAME
rusers – who is logged in on remote machines

SYNOPSIS
rusers [-ahilu] host...

DESCRIPTION
The rusers command produces output similar to who(1), but for remote machines.
The listing is in the order that responses are received, but this order can be changed by
specifying one of the options listed below.

The default is to print out the names of the users logged in. When the -l flag is given,
additional information is printed for each user:

userid hostname : terminal login_date login_time idle_time login_host

If hostname and login host are the same value, the login_host field is not displayed.
Likewise, if hostname is not idle, the idle_time is not displayed.

A remote host will only respond if it is running the rusersd daemon, which may be
started up from inetd(1M) or listen(1M).

OPTIONS
-a Give a report for a machine even if no users are logged on.
-h Sort alphabetically by host name.
-i Sort by idle time.
-l Give a longer listing in the style of who(1).
-u Sort by number of users.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWrcmdc</td>
</tr>
</tbody>
</table>

SEE ALSO
who(1), inetd(1M), listen(1M), padmin(1M), sacadm(1M), attributes(5)
rwho(1)

NAME rwho – who is logged in on local machines

SYNOPSIS rwho [-a]

DESCRIPTION The rwho command produces output similar to who(1), but for all machines on your network. If no report has been received from a machine for 5 minutes, rwho assumes the machine is down, and does not report users last known to be logged into that machine.

If a user has not typed to the system for a minute or more, rwho reports this idle time. If a user has not typed to the system for an hour or more, the user is omitted from the output of rwho unless the -a flag is given.

OPTIONS -a Report all users whether or not they have typed to the system in the past hour.

FILES /var/spool/rwho/whod.* information about other machines

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWrcmds</td>
</tr>
</tbody>
</table>

SEE ALSO finger(1), uptime(1), who(1), in.rwhod(1M), attributes(5)

NOTES rwho does not work through gateways.

The directory /var/spool/rwho must exist on the host from which rwho is run.

This service takes up progressively more network bandwidth as the number of hosts on the local net increases. For large networks, the cost becomes prohibitive.

The rwho service daemon, in.rwhod(1M), must be enabled for this command to return useful results.
NAME
sag – system activity graph

SYNOPSIS

DESCRIPTION
The sag utility graphically displays the system activity data stored in a binary data file by a previous sar(1) run. Any of the sar data items may be plotted singly or in combination, as cross plots or versus time. Simple arithmetic combinations of data may be specified. sag invokes sar and finds the desired data by string-matching the data column header (run sar to see what is available). The sag utility requires a graphic terminal to draw the graph, and uses tplot(1) to produce its output. When running Solaris 2.X and OpenWindows, perform the following steps:

1. Run an "xterm" as a Tektronics terminal: prompt# xterm -t
2. In the "xterm" window, run sag specifying a tek terminal: prompt# sag -T tek

OPTIONS
The following options are supported and passed through to sar (see sar(1)):

-e time Select data up to time. Default is 18:00.
-f file Use file as the data source for sar. Default is the current daily data file /usr/adm/sa/sadd.
-i sec Select data at intervals as close as possible to sec seconds.
-s time Select data later than time in the form hh [mm]. Default is 08:00.
-T term Produce output suitable for terminal term. See tplot(1) for known terminals. Default for term is $TERM.
-x spec x axis specification with spec in the form:
name [op name]...[lo hi]

name is either a string that will match a column header in the sar report, with an optional device name in square brackets, for example, r+w/s[dsk-1], or an integer value. op is + - * / surrounded by blank spaces. Up to five names may be specified. Parentheses are not recognized. Contrary to custom, + and − have precedence over * and /. Evaluation is left to right. Thus, A/A+B*100 is evaluated as (A/(A+B))*100, and A+B/C+D is (A+B)/(C+D). lo and hi are optional numeric scale limits. If unspecified, they are deduced from the data.

Enclose spec in double-quotes (" ") if it includes white space.

A single spec is permitted for the x axis. If unspecified, time is used.

-y spec y axis specification with spec in the same form as for -x. Up to 5 spec arguments separated by a semi-colon (;) may be given for -y. The -y default is:

-y"%usr0100;%usr+%sys0100;%usr+%sys+%wio0100"
EXAMPLES

EXAMPLE 1  Examples of the sag command.

To see today’s CPU utilization:

example$ sag

To see activity over 15 minutes of all disk drives:

example$ TS=`date +%H:%M`
example$ sar -o /tmp/tempfile 60 15
example$ TE=`date +%H:%M`
example$ sag -f /tmp/tempfile -s $TS -e $TE -y "r+w/s[dsk]"

FILES
/usr/adm/sa/sadd  daily data file for day dd

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWaccu</td>
</tr>
</tbody>
</table>

SEE ALSO  sar(1), tplot(1), attributes(5)
NAME
sar – system activity reporter

SYNOPSIS
sar [-aAbcdgkmpqruvw] [-o filename] t [n]
sar [-aAbcdgkmpqruvw] [-e time] [-f filename] [-i sec] [-s time]

DESCRIPTION
In the first instance, sar samples cumulative activity counters in the operating system at n intervals of t seconds, where t should be 5 or greater. If t is specified with more than one option, all headers are printed together and the output may be difficult to read. (If the sampling interval is less than 5, the activity of sar itself may affect the sample.) If the -o option is specified, it saves the samples in filename in binary format. The default value of n is 1.

In the second instance, no sampling interval is specified. sar extracts data from a previously recorded filename, either the one specified by the -E option or, by default, the standard system activity daily data file /var/adm/sa/sadd for the current day dd. The starting and ending times of the report can be bounded using the -e and -s arguments with time specified in the form hh:mm:ss. The -i option selects records at sec second intervals. Otherwise, all intervals found in the data file are reported.

OPTIONS
The following options modify the subsets of information reported by sar.

-a Report use of file access system routines: iget/s, namei/s, dirblk/s
-A Report all data. Equivalent to -abcdgkmpqruvw.
-b Report buffer activity:
  bread/s, bwrit/s
  transfers per second of data between system buffers and disk or other block devices.
  lread/s, lwrit/s
  accesses of system buffers.
  %rcache, %wcache
  cache hit ratios, that is, (1–bread/lread) as a percentage.
-sc Report system calls:
  scall/s
  system calls of all types.
  sread/s, swrit/s, fork/s, exec/s
  specific system calls.
  rchar/s, wchar/s
  characters transferred by read and write system calls. No incoming or outgoing exec(2) and fork(2) calls are reported.
-d Report activity for each block device (for example, disk or tape drive) with the exception of XDC disks and tape drives. When
data is displayed, the device specification *disk-* is generally used to represent a disk drive. The device specification used to represent a tape drive is machine dependent. The activity data reported is:

- **%busy, avque**  
  portion of time device was busy servicing a transfer request, average number of requests outstanding during that time.

- **read/s, write/s, blks/s**  
  number of read/write transfers from or to device, number of bytes transferred in 512-byte units.

- **avwait**  
  average wait time in milliseconds.

- **avserv**  
  average service time in milliseconds.

For more general system statistics, use `iostat(1M)`, `sar(1M)`, or `vmstat(1M)`.

See *System Administration Guide: Basic Administration* for naming conventions for disks.

- **-e time**  
  Select data up to `time`. Default is 18:00.

- **-f filename**  
  Use `filename` as the data source for `sar`. Default is the current daily data file `/var/adm/`.*sa/`*saad`.

- **-g**  
  Report paging activities:

  - **pgout/s**  
    page-out requests per second.

  - **ppgout/s**  
    pages paged-out per second.

  - **pgfree/s**  
    pages per second placed on the free list by the page stealing daemon.

  - **pgscan/s**  
    pages per second scanned by the page stealing daemon.

  - **%ufs_ipf**  
    the percentage of UFS inodes taken off the freelist by `iget` which had reusable pages associated with them. These pages are flushed and cannot be reclaimed by processes. Thus, this is the percentage of `igets` with page flushes.

- **-i sec**  
  Select data at intervals as close as possible to `sec` seconds.

- **-k**  
  Report kernel memory allocation (KMA) activities:

  - **sml_mem, alloc, fail**  
    information about the memory pool reserving and allocating space for small requests: the amount of memory in bytes KMA
has for the small pool, the number of bytes allocated to satisfy requests for small amounts of memory, and the number of requests for small amounts of memory that were not satisfied (failed).

lg_mem, alloc, fail
information for the large memory pool (analogous to the information for the small memory pool).

ovsz_alloc, fail
the amount of memory allocated for oversize requests and the number of oversize requests which could not be satisfied (because oversized memory is allocated dynamically, there is not a pool).

-m Report message and semaphore activities:
msg/s, sema/s primitives per second.

-o filename Save samples in file, filename, in binary format.

-p Report paging activities:
atch/s page faults per second that are satisfied by reclaiming a page currently in memory (attaches per second).
pgin/s page-in requests per second.
ppgin/s pages paged-in per second.
pflt/s page faults from protection errors per second (illegal access to page) or "copy-on-writes".
vflt/s address translation page faults per second (valid page not in memory).
slock/s faults per second caused by software lock requests requiring physical I/O.

-q Report average queue length while occupied, and percent of time occupied:
runq-sz, %runocc run queue of processes in memory and runnable.
swpq-sz, %swpocc these are no longer reported by sar.

-r Report unused memory pages and disk blocks:
fremem average pages available to user processes.
freeswap disk blocks available for page swapping.
sar(1)

- s time Select data later than time in the form hh:mm. Default is 08:00.
- u Report CPU utilization (the default):
  %usr, %sys, %wio, %idle
  portion of time running in user mode, running in system mode,
  idle with some process waiting for block I/O, and otherwise
  idle.
- v Report status of process, i-node, file tables:
  proc-sz, inod-sz, file-sz, lock-sz
  entries/size for each table, evaluated once at sampling point.
  ov
  overflows that occur between sampling points for each table.
- w Report system swapping and switching activity:
  swpin/s, swpot/s, bswin/s, bswot/s
  number of transfers and number of 512-byte units transferred
  for swaps and swapouts (including initial loading of some
  programs).
  pswch/s
  process switches.
- y Report TTY device activity:
  rawch/s, canch/s, outch/s
  input character rate, input character rate processed by canon,
  output character rate.
  rcvin/s, xmtin/s, mdmin/s
  receive, transmit and modem interrupt rates.

EXAMPLES

EXAMPLE 1 Viewing system activity
To see today’s CPU activity so far:
example% sar

EXAMPLE 2 Watching system activity evolve
To watch CPU activity evolve for 10 minutes and save data:
example% sar -o temp 60 10

EXAMPLE 3 Reviewing disk and tape activity
To later review disk and tape activity from that period:
example% sar -d -f temp

FILES
/var/adm/sa/sadd daily data file, where dd are digits representing the day
of the month

1298 man pages section 1: User Commands • Last Revised 25 Oct 2001
See attributes(5) for descriptions of the following attributes:

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWaccu</td>
</tr>
</tbody>
</table>

SEE ALSO sag(1), iostat(1M), sar(1M), vmstat(1M), exec(2), fork(2), attributes(5)

NOTES The sum of CPU utilization might vary slightly from 100 because of rounding errors in the production of a percentage figure.
sccs(1)

NAME  sccs – front end for the Source Code Control System (SCCS)

SYNOPSIS  
/usr/ccs/bin/sccs  [-r] [-d rootprefix] [-p subdir] subcommand [option...]  
[/file...]

/usr/xpg4/bin/sccs  [-r] [-d rootprefix] [-p subdir] subcommand [option...]  
[/file...]

DESCRIPTION  The sccs command is a comprehensive, straightforward front end to the various utility programs of the Source Code Control System (SCCS).

sccs applies the indicated subcommand to the history file associated with each of the indicated files.

The name of an SCCS history file is derived by prepending the ‘s.’ prefix to the filename of a working copy. The sccs command normally expects these ‘s.files’ to reside in an SCCS subdirectory. Thus, when you supply sccs with a file argument, it normally applies the subcommand to a file named s.file in the SCCS subdirectory. If file is a path name, sccs looks for the history file in the SCCS subdirectory of that file’s parent directory. If file is a directory, however, sccs applies the subcommand to every s.file it contains. Thus, the command:

test% sccs get program.c

would apply the get subcommand to a history file named SCCS/s.program.c, while the command:

test% sccs get SCCS

would apply it to every s.file in the SCCS subdirectory.

Options for the sccs command itself must appear before the subcommand argument. Options for a given subcommand must appear after the subcommand argument. These options are specific to each subcommand, and are described along with the subcommands themselves (see Subcommands below).

Running Setuid  The sccs command also includes the capability to run “setuid” to provide additional protection. However, this does not apply to subcommands such as sccs-admin(1), since this would allow anyone to change the authorizations of the history file. Commands that would do so always run as the real user.

OPTIONS  The following options are supported:

/options/ccs/bin/sccs  -d rootprefix

/options/xpg4/bin/sccs  -d rootprefix

Define the root portion of the path name for SCCS history files. The default root portion is the current directory. rootprefix is prepended to the entire file argument, even if file is an absolute path name. -d overrides any directory specified by the PROJECTDIR environment variable (see ENVIRONMENT VARIABLES below).
Define the (sub)directory within which a history file is expected to reside. SCCS is the default. (See EXAMPLES, below).

- `sccs` with the real user ID, rather than set to the effective user ID.

**OPERANDS**

The following operands are supported:

- `subcommand` An SCCS utility name or the name of one of the pseudo-utilities listed in `USAGE`.
- `options` An option or option-argument to be passed to `subcommand`.
- `operands` An operand to be passed to `subcommand`.

**Subcommands**

Many of the following `sccs` subcommands invoke programs that reside in `/usr/ccs/bin`. Many of these subcommands accept additional arguments that are documented in the reference page for the utility program the subcommand invokes.

- `/usr/ccs/bin/sccs` `cdc -rsid [-y][comment]]`
- `/usr/xpg4/bin/sccs` `cdc -rsid | -rsid [-y][comment]]`

Annotate (change) the delta commentary. Refer to `sccs-cdc(1)`. The `fix` subcommand can be used to replace the delta, rather than merely annotating the existing commentary.

- `/usr/ccs/bin/sccs` `-rsid`
- `/usr/xpg4/bin/sccs` `-r sid` `-rsid`

Specify the SCCS delta ID (SID) to which the change notation is to be added. The SID for a given delta is a number, in Dewey decimal format, composed of two or four fields: the `release` and `level` fields, and for branch deltas, the `branch` and `sequence` fields. For instance, the SID for the initial delta is normally `1.1`.

- `-y[comment]` Specify the comment with which to annotate the delta commentary. If `-y` is omitted, `sccs` prompts for a comment. A null comment results in an empty annotation.

- `/usr/ccs/bin/sccs` `check [-b] [-u[username]]`
- `/usr/xpg4/bin/sccs` `check [-b] [-u [username] | -U ]`

Check for files currently being edited. Like `info` and `tell`, but returns an exit code, rather than producing a listing of files. `check` returns a non-zero exit status if anything is being edited.
**sccs(1)**

- **-b** Ignore branches.
- **-u[username]**
- **-u [ username] | -U** Check only files being edited by you. When **username** is specified, check only files being edited by that user. For **/usr/xpg4/bin/sccs**, the **-U** option is equivalent to **-u <current_user>**.

**clean [-b]**
Remove everything in the current directory that can be retrieved from an SCCS history. Does not remove files that are being edited.

- **-b** Do not check branches to see if they are being edited. ‘**clean -b**’ is dangerous when branch versions are kept in the same directory.

**comb**
Generate scripts to combine deltas. Refer to **sccs-comb(1)**.

**create**
Create (initialize) history files. **create** performs the following steps:
- Renames the original source file to **.program.c** in the current directory.
- Create the history file called **s.program.c** in the SCCS subdirectory.
- Performs an ‘**sccs get**’ on **program.c** to retrieve a read-only copy of the initial version.

**deledit [-s] [-y[comment]]**
Equivalent to an ‘**sccs delta**’ and then an ‘**sccs edit**’. **deledit** checks in a delta, and checks the file back out again, but leaves the current working copy of the file intact.

- **-s** Silent. Do not report delta numbers or statistics.
- **-y [comment]**
  Supply a comment for the delta commentary. If **-y** is omitted, delta prompts for a comment. A null comment results in an empty comment field for the delta.

**delget [-s] [-y[comment]]**
Perform an ‘**sccs delta**’ and then an ‘**sccs get**’ to check in a delta and retrieve read-only copies of the resulting new version. See the **deledit** subcommand for a description of **-s** and **-y**. **sccs** performs a **delta** on all the files specified in the argument list, and then a **get** on all the files. If an error occurs during the **delta**, the **get** is not performed.

**delta [-s] [-y[comment]]**
Check in pending changes. Records the line-by-line changes introduced while the file was checked out. The effective user ID must be the same as the ID of the person who has the file checked out. Refer to **sccs-delta(1)**. See the **deledit** subcommand for a description of **-s** and **-y**.
Compare (in diff(1) format) the working copy of a file that is checked out for editing, with a version from the SCCS history. Use the most recent checked-in version by default. The diffs subcommand accepts the same options as diff.

Any -r, -c, -i, -x, and -t options are passed to subcommand get. A -c option is passed to diff as -c. An -I option is passed to diff as -i.

- cdate-time

Use the most recent version checked in before the indicated date and time for comparison. date-time takes the form: yy[mm][dd][hh][mm][ss]]. Omitted units default to their maximum possible values; that is -c7502 is equivalent to -c750228235959.

- r sid | - rsid

Use the version corresponding to the indicated delta for comparison.

edit

Retrieve a version of the file for editing. 'sccs edit' extracts a version of the file that is writable by you, and creates a p. file in the SCCS subdirectory as lock on the history, so that no one else can check that version in or out. ID keywords are retrieved in unexpanded form. edit accepts the same options as get, below. Refer to sccs-get(1) for a list of ID keywords and their definitions.

enter

Similar to create, but omits the final 'sccs get'. This may be used if an 'sccs edit' is to be performed immediately after the history file is initialized.

fix - rsid

Revise a (leaf) delta. Remove the indicated delta from the SCCS history, but leave a working copy of the current version in the directory. This is useful for incorporating trivial updates for which no audit record is needed, or for revising the delta commentary. fix must be followed by a -r option, to specify the SID of the delta to remove. The indicated delta must be the most recent (leaf) delta in its branch. Use fix with caution since it does not leave an audit trail of differences (although the previous commentary is retained within the history file).
Retrieve a version from the SCCS history. By default, this is a read-only working copy of the most recent version; ID keywords are in expanded form. Refer to sccs-get(1), which includes a list of ID keywords and their definitions.

- \(-\,e\) Retrieve a version for editing. Same as sccs edit.

- \(-G\) newname | -G newname Use newname as the name of the retrieved version.

- \(-k\) Retrieve a writable copy but do not check out the file. ID keywords are unexpanded.

- \(-m\) Precede each line with the SID of the delta in which it was added.

- \(-p\) Produce the retrieved version on the standard output. Reports that would normally go to the standard output (delta IDs and statistics) are directed to the standard error.

- \(-s\) Silent. Do not report version numbers or statistics.

- \(-c\) date-time | -c date-time Retrieve the latest version checked in prior to the date and time indicated by the date-time argument. date-time takes the form: yy[mm[dd[ hh[mm:ss]]]]

- \(-r\) sid | -r sid Retrieve the version corresponding to the indicated SID. If no sid is specified, the latest sid for the specified file is retrieved.

help
message-code \(\mid\) sccs-command

help stuck Supply more information about SCCS diagnostics. help displays a brief explanation of the error when you supply the code displayed by an SCCS diagnostic message. If you supply the name of an SCCS command, it prints a usage line. help also recognizes the keyword stuck. Refer to sccs-help(1).

info \(-b\) \(-u[username]\) Display a list of files being edited, including the version number checked out, the version to be checked in, the name of the user who holds the lock, and the date and time the file was checked out.

- \(-b\) Ignore branches.
### User Commands

**sccs(1)**

- **/usr/ccs/bin/sccs**
  - `-u[username]`
  - List only files checked out by you. When `username` is specified, list only files checked out by that user. For `/usr/xpg4/bin/sccs`, the `-U` option is equivalent to `-u current_user`.

- **/usr/xpg4/bin/sccs**
  - `-u [username]` | `-U`

**print**

- **/usr/ccs/bin/sccs**
  - `print`
  - Print the entire history of each named file. Equivalent to an `sccs prs -e` followed by an `sccs get -p -m`.

- **/usr/xpg4/bin/sccs**
  - `print`

**prs [-e] [-cdate-time] [-rsid]**

- **/usr/ccs/bin/sccs**
  - `prs [-e] [-cdate-time] [-rsid]`
  - Peruse (display) the delta table, or other portion of an s. file. Refer to `sccs-prs(1)`.

  - `-e`
    - Display delta table information for all deltas earlier than the one specified with `-r` (or all deltas if none is specified).

  - `-l`
    - Display information for all deltas later than, and including, that specified by `-c` or `-r`.

- **/usr/xpg4/bin/sccs**
  - `prs [-e] [-c ddate-time | -cdate-time] [-rsid | -r sid]`

**prt [-y]**

- **/usr/ccs/bin/sccs**
  - `prt [-y]`
  - Display the delta table, but omit the MR field (see `sccsfile(4)` for more information on this field). Refer to `sccs-prt(1)`.

  - `-y`
    - Display the most recent delta table entry. The format is a single output line for each file argument, which is convenient for use in a pipeline with `awk(1)` or `sed(1)`.

- **/usr/xpg4/bin/sccs**
  - `prt [-y]`

**rmdel -rsid**

- **/usr/ccs/bin/sccs**
  - `rmdel -rsid`
  - Remove the indicated delta from the history file. That delta must be the most recent (leaf) delta in its branch. Refer to `sccs-rmdel(1)`.

- **/usr/xpg4/bin/sccs**
  - `rmdel -r sid`

**sact**

- **/usr/ccs/bin/sccs**
  - `sact`
  - Show editing activity status of an SCCS file. Refer to `sccs-sact(1)`.

- **/usr/xpg4/bin/sccs**
  - `sact`

**sccsdiff -r old-sid -r new-sid diff-options**

- **/usr/ccs/bin/sccs**
  - `sccsdiff -r old-sid -r new-sid diff-options`
  - Compare two versions corresponding to the indicated SIDs (deltas) using `diff`. Refer to `sccs-sccsdiff(1)`.

- **/usr/xpg4/bin/sccs**
  - `sccsdiff -r old-sid -r new-sid diff-options`

**tell [-b] [-u[username]]**

- **/usr/ccs/bin/sccs**
  - `tell [-b] [-u[username]]`
  - Display the list of files that are currently checked out, one file per line.

- **/usr/xpg4/bin/sccs**
  - `tell [-b] [-u[username]]` | `-U`
Ignore branches.

List only files checked out to you. When `username` is specified, list only files checked out to that user. For `/usr/xpg4/bin/sccs`, the `-U` option is equivalent to `-u <current_user>`.

“Undo” the last edit or `get -e`, and return the working copy to its previous condition. `unedit` backs out all pending changes made since the file was checked out.

Same as `unedit`. Refer to `sccs-unget(1)`.

Validate the history file. Refer to `sccs-val(1)`.

Display any expanded ID keyword strings contained in a binary (object) or text file. Refer to `what(1)` for more information.

EXAMPLE 1 Checking out, editing, and checking in a file

To check out a copy of `program.c` for editing, edit it, and then check it back in:

```
example% sccs edit program.c
1.1 new delta 1.2
14 lines

example% vi program.c
your editing session

example% sccs delget program.c
comments? clarified cryptic diagnostic
1.2
3 inserted
2 deleted
12 unchanged
1.2
15 lines
```

EXAMPLE 2 Defining the root portion of the command pathname

`sccs` converts the command:

```
example% sccs -d/usr/src/include get stdio.h
```

to:

```
/usr/ccs/bin/get  /usr/src/include/SCCS/sstdio.h
```
EXAMPLE 3  Defining the resident subdirectory

The command:

```
example% sccs -pprivate get include/stdio.h
```

becomes:

```
/usr/ccs/bin/get include/private/s_stdio.h
```

EXAMPLE 4  Initializing a history file

To initialize the history file for a source file named `program.c`, make the SCCS subdirectory, and then use `sccs create`:

```
exmple$ mkdir SCCS
example$ sccs create program.c
program.c:
1.1
14 lines
```

After verifying the working copy, you can remove the backup file that starts with a comma:

```
exmple$ diff program.c ,program.c
example$ rm ,program.c
```

EXAMPLE 5  Retrieving a file from another directory

To retrieve a file from another directory into the current directory:

```
exmple$ sccs get /usr/src/sccs/cc.c
```

or:

```
exmple$ sccs -p/usr/src/sccs/ get cc.c
```

EXAMPLE 6  Checking out all files

To check out all files under SCCS in the current directory:

```
exmple$ sccs edit SCCS
```

EXAMPLE 7  Checking in all files

To check in all files currently checked out to you:

```
exmple$ sccs delta 'sccs tell -u'
```
**EXAMPLE 8** Entering multiple lines of comments

If using `-y` to enter a comment, for most shells, enclose the comment in single or double quotes. In the following example, `Myfile` is checked in with a two-line comment:

```bash
example% sccs deledit Myfile -y"Entering a multi-line comment"
```

No id keywords (cm7)

1.2

2 inserted

0 deleted

14 unchanged

1.2

new delta 1.3

Displaying the SCCS history of `Myfile`:

```bash
example% sccs prt Myfile
```

SCCS/s.Myfile:

```
D 1.2 01/04/20 16:37:07 me 2 1 00002/00000/00014
Entering a multi-line comment
```

```
D 1.1 01/04/15 13:23:32 me 1 0 00014/00000/00000
date and time created 01/04/15 13:23:32 by me
```

If `-y` is not used and `sccs` prompts for a comment, the newlines must be escaped using the backslash character (`\`):

```bash
example% sccs deledit Myfile
```

```
comments? Entering a \multi-line comment
```

No id keywords (cm7)

1.2

0 inserted

0 deleted

14 unchanged

1.2

new delta 1.3

**ENVIRONMENT VARIABLES**

See `environ(5)` for descriptions of the following environment variables that affect the execution of `sccs`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

**PROJECTDIR**

If contains an absolute path name (beginning with a slash), `sccs` searches for SCCS history files in the directory given by that variable.

If `PROJECTDIR` does not begin with a slash, it is taken as the name of a user, and `sccs` searches the `src` or `source` subdirectory of that user’s home directory for history files. If such a directory is found, it is used. Otherwise, the value is used as a relative path.
name.

EXIT STATUS

The following exit values are returned:

   0  Successful completion.
   >0  An error occurred.

FILES

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCCS</td>
<td>SCCS subdirectory</td>
</tr>
<tr>
<td>SCCS/a/file</td>
<td>temporary file of differences</td>
</tr>
<tr>
<td>SCCS/p/file</td>
<td>lock (permissions) file for checked-out versions</td>
</tr>
<tr>
<td>SCCS/q/file</td>
<td>temporary file</td>
</tr>
<tr>
<td>SCCS/s/file</td>
<td>SCCS history file</td>
</tr>
<tr>
<td>SCCS/x/file</td>
<td>temporary copy of the s.file</td>
</tr>
<tr>
<td>SCCS/z/file</td>
<td>temporary lock file</td>
</tr>
<tr>
<td>/usr/ccs/bin/*</td>
<td>SCCS utility programs</td>
</tr>
</tbody>
</table>

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>File</th>
<th>Attribute Type</th>
<th>Attribute Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>/usr/ccs/bin/sccs</td>
<td>Availability</td>
<td>SUNWsprot</td>
</tr>
<tr>
<td>/usr/xpg4/bin/sccs</td>
<td>Availability</td>
<td>SUNWxcu4t</td>
</tr>
</tbody>
</table>

SEE ALSO

awk(1), diff(1), sccs-admin(1), sccs-cdc(1), sccs-cmd(1), sccs-comb(1), sccs-delta(1),
sccs-get(1), sccs-help(1), sccs-prs(1), sccs-rmdel(1), sccs-sact(1),
sccs-sccsdiff(1), sccs-unget(1), sccs-val(1), sed(1), what(1), sccsfile(4),
attributes(5), XPG4(5)
NAME  sccs-admin, admin – create and administer SCCS history files

SYNOPSIS  /usr/ccs/bin/admin [-bhnz] [-a username | groupid...] [-d flag] ...
            [-e username | groupid...] [-f flag [value]] ... [-i [filename]]
            [-m mr-list] [-r release] [-t [description-file]] [-y [comment]] s.filename...

DESCRIPTION  The admin command creates or modifies the flags and other parameters of SCCS
              history files. Filenames of SCCS history files begin with the ‘s.’ prefix, and are
              referred to as s.files, or “history” files.

              The named s.file is created if it does not exist already. Its parameters are initialized or
              modified according to the options you specify. Parameters not specified are given
              default values when the file is initialized, otherwise they remain unchanged.

              If a directory name is used in place of the s.filename argument, the admin command
              applies to all s.files in that directory. Unreadable s.files produce an error. The use of
              ‘−’ as the s.filename argument indicates that the names of files are to be read from the
              standard input, one s.file per line.

OPTIONS  The following options are supported:

    −b
              Force encoding of binary data. Files that contain ASCII NUL or other control
              characters, or that do not end with a NEWLINE, are recognized as binary data files.
              The contents of such files are stored in the history file in encoded form. See
              uuencode(1C) for details about the encoding. This option is normally used in
              conjunction with −i to force admin to encode initial versions not recognized as
              containing binary data.

    −h
              Check the structure of an existing s.file (see sccsfile(4)), and compare a newly
              computed check-sum with one stored in the first line of that file. −h inhibits writing
              on the file; and so nullifies the effect of any other options.

    −n
              Create a new SCCS history file.

    −z
              Recompute the file check-sum and store it in the first line of the s.file. Caution: it is
              important to verify the contents of the history file (see sccs-val(1), and the print
              subcommand in sccs(1)), since using −z on a truly corrupted file may prevent
              detection of the error.

    −a username | groupid
              Add a user name, or a numerical group ID, to the list of users who may check
              deltas in or out. If the list is empty, any user is allowed to do so.

    −d flag
              Delete the indicated flag from the SCCS file. The −d option may be specified only
              for existing s.files. See −f for the list of recognized flags.
Erase a user name or group ID from the list of users allowed to make deltas.

Set the indicated flag to the (optional) value specified. The following flags are recognized:

-e username  Group id
Erase a user name or group ID from the list of users allowed to make deltas.

-\f flag  [value]
Set a ceiling on the releases that can be checked out. \f is a number less than or equal to 9999. If \c is not set, the ceiling is 9999.

Set a floor on the releases that can be checked out. The floor is a number greater than 0 but less than 9999. If \f is not set, the floor is 1.

The default delta number, or SID, to be used by an SCCS get command.

 Treat the ‘No id keywords (ge6)’ message issued by an SCCS get or delta command as an error rather than a warning.

 Allow concurrent updates.

 Lock the indicated list of releases against deltas. If \a is used, lock out deltas to all releases. An SCCS ‘get -e’ command fails when applied against a locked release.

 Create empty releases when releases are skipped. These null (empty) deltas serve as anchor points for branch deltas.

 Supply a value to which the keyword is to expand when a read-only version is retrieved with the SCCS get command.

 Supply a value for the module name to which the sccs-admin.1 keyword is to expand. If the m flag is not specified, the value assigned is the name of the SCCS file with the leading s. removed.

 Supply a value for the module type to which the keyword is to expand.

 Specify a validation program for the MR numbers associated with a new delta. The optional program specifies the name of an MR number validity.
checking program. If this flag is set when creating an SCCS file, the -m option must also be used, in which case the list of MRs may be empty.

- i [filename]
  Initialize the history file with text from the indicated file. This text constitutes the initial delta, or set of checked-in changes. If filename is omitted, the initial text is obtained from the standard input. Omitting the -i option altogether creates an empty s.file. You can only initialize one s.file with text using -i. This option implies the -n option.

- m mr-list
  Insert the indicated Modification Request (MR) numbers into the commentary for the initial version. When specifying more than one MR number on the command line, mr-list takes the form of a quoted, space-separated list. A warning results if the v flag is not set or the MR validation fails.

- r release
  Specify the release for the initial delta. -r may be used only in conjunction with -i. The initial delta is inserted into release 1 if this option is omitted. The level of the initial delta is always 1; initial deltas are named 1.1 by default.

- t [description-file]
  Insert descriptive text from the file description-file. When -t is used in conjunction with -n, or -i to initialize a new s.file, the description-file must be supplied. When modifying the description for an existing file: a -t option without a description-file removes the descriptive text, if any; a -t option with a description-file replaces the existing text.

- y [comment]
  Insert the indicated comment in the “Comments:” field for the initial delta. Valid only in conjunction with -i or -n. If -y option is omitted, a default comment line is inserted that notes the date and time the history file was created.

EXIT STATUS
  The following exit values are returned:
  0  Successful completion.
  1  An error occurred.

FILES
  s.*  history file
  SCCS/s.*  history file in SCCS subdirectory
  z.*  temporary lock file

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsprot</td>
</tr>
</tbody>
</table>
Use the SCCS help command for explanations (see sccs-help(1)).

The last component of all SCCS filenames must have the ‘s.’ prefix. New SCCS files are given mode 444 (see chmod(1)). All writing done by admin is to a temporary file with an x. prefix, created with mode 444 for a new SCCS file, or with the same mode as an existing SCCS file. After successful execution of admin, the existing s. file is removed and replaced with the x. file. This ensures that changes are made to the SCCS file only when no errors have occurred.

It is recommended that directories containing SCCS files have permission mode 755, and that the s. files themselves have mode 444. The mode for directories allows only the owner to modify the SCCS files contained in the directories, while the mode of the s. files prevents all modifications except those performed using SCCS commands.

If it should be necessary to patch an SCCS file for any reason, the mode may be changed to 644 by the owner to allow use of a text editor. However, extreme care must be taken when doing this. The edited file should always be processed by an ‘admin -h’ to check for corruption, followed by an ‘admin -z’ to generate a proper check-sum. Another ‘admin -h’ is recommended to ensure that the resulting s. file is valid.

admin also uses a temporary lock s. file, starting with the ‘z.’ prefix, to prevent simultaneous updates to the s. file. See sccs-get(1) for further information about the ‘z. file’.

SEE ALSO sccs(1), sccs-cdc(1), sccs-delta(1), sccs-get(1), sccs-help(1), sccs-rmdel(1), sccs-val(1), sccsfile(4), attributes(5)

DIAGNOSTICS

WARNINGS

User Commands  1313
NAME
sccs-cdc, cdc – change the delta commentary of an SCCS delta

SYNOPSIS
/usr/ccs/bin/cdc -r sid [-m mr-list] [-y [comment]] s.filename...

DESCRIPTION
cdc annotates the delta commentary for the SCCS delta ID (SID) specified by the -r option in each named .s.file.

If the v flag is set in the .s.file, you can also use cdc to update the Modification Request (MR) list.

If you checked in the delta, or, if you own the file and directory and have write permission, you can use cdc to annotate the commentary.

Rather than replacing the existing commentary, cdc inserts the new comment you supply, followed by a line of the form:

*** CHANGED *** yy/mm/dd hh/mm/ss username

above the existing commentary.

If a directory is named as the s.filename argument, the cdc command applies to all .s.files in that directory. Unreadable .s.files produce an error; processing continues with the next file (if any). If `-.' is given as the s.filename argument, each line of the standard input is taken as the name of an SCCS history file to be processed, and the -m and -y options must be used.

OPTIONS
- r sid Specify the SID of the delta to change.
- m mr-list Specify one or more MR numbers to add or delete. When specifying more than one MR on the command line, mr-list takes the form of a quoted, space-separated list. To delete an MR number, precede it with a ! character (an empty MR list has no effect). A list of deleted MRs is placed in the comment section of the delta commentary. If -m is not used and the standard input is a terminal, cdc prompts with MRs? for the list (before issuing the comments? prompt). -m is only useful when the v flag is set in the .s.file. If that flag has a value, it is taken to be the name of a program to validate the MR numbers. If that validation program returns a non-zero exit status, cdc terminates and the delta commentary remains unchanged.
- y[comment] Use comment as the annotation in the delta commentary. The previous comments are retained; the comment is added along with a notation that the commentary was changed. A null comment leaves the commentary unaffected. If -y is not specified and the standard input is a terminal, cdc prompts with comments? for the text of the notation to be added. An unescaped NEWLINE character terminates the annotation text.
EXAMPLE 1 Changing the annotated commentary

The following command:

```
example% cdc -r1.6 -y"corrected commentary" s.program.c
```

produces the following annotated commentary for delta 1.6 in `s.program.c`:

```
D 1.6 88/07/05 23:21:07 username 9 0 00001/00000/00000
MRs:
COMMENTS:
corrected commentary
*** CHANGED *** 88/07/07 14:09:41 username
performance enhancements in main()
```

FILES `z.file` temporary lock file

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsprot</td>
</tr>
</tbody>
</table>

SEE ALSO `sccs(1), sccs-admin(1), sccs-comb(1), sccs-delta(1), sccs-help(1), sccs-prs(1), sccs-prt(1), sccs-rmdel(1), what(1), sccsfile(4), attributes(5)`

DIAGNOSTICS Use the SCCS help command for explanations (see `sccs-help(1)`).
NAME sccs-comb, comb – combine SCCS deltas

SYNOPSIS /usr/ccs/bin/comb [-os] [-csid-list] [-psid] s.filename...

DESCRIPTION comb generates a shell script (see sh(1)) that you can use to reconstruct the indicated s. files. This script is written to the standard output.

If a directory name is used in place of the s.filename argument, the comb command applies to all s. files in that directory. Unreadable s. files produce an error; processing continues with the next file (if any). The use of ‘−’ as the s.filename argument indicates that the names of files are to be read from the standard input, one s. file per line.

If no options are specified, comb preserves only the most recent (leaf) delta in a branch, and the minimal number of ancestors needed to preserve the history.

OPTIONS The following options are supported:

- o For each ‘get -e’ generated, access the reconstructed file at the release of the delta to be created. Otherwise, the reconstructed file is accessed at the most recent ancestor. The use of o may decrease the size of the reconstructed s. file. It may also alter the shape of the delta tree of the original file.

- s Generate scripts to gather statistics, rather than combining deltas. When run, the shell scripts report: the file name, size (in blocks) after combining, original size (also in blocks), and the percentage size change, computed by the formula:

\[100 \times \frac{\text{original} - \text{combined}}{\text{original}}\]

This option can be used to calculate the space that will be saved, before actually doing the combining.

- csid-list Include the indicated list of deltas. All other deltas are omitted. sid-list is a comma-separated list of SCCS delta IDs (SIDs). To specify a range of deltas, use a ‘−’ separator instead of a comma, between two SIDs in the list.

- psID The SID of the oldest delta to be preserved.

FILES s. COMB reconstructed SCCS file
comb????? temporary file

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsprot</td>
</tr>
</tbody>
</table>
Use the SCCS help command for explanations (see sccs-help(1)).

**BUGS**

`comb` may rearrange the shape of the tree of deltas. It may not save any space; in fact, it is possible for the reconstructed file to actually be larger than the original.
sccs-delta, delta – make a delta to an SCCS file

SYNOPSIS

```
/usr/ccs/bin/delta [-d] [-n] [-p] [-s] [-g sid-list] [-m mr-list] [-y comment] s.filename...
```

```
/usr/xpg4/bin/delta [-d] [-n] [-p] [-s] [-g sid-list] [-m mr-list] [-y comment] s.filename...
```

DESCRIPTION

delta checks a record of the line-by-line differences made to a checked-out version of a file under SCCS control. These changes are taken from the writable working copy that was retrieved using the SCCS get command (see `sccs-get(1)`). This working copy does not have the 's.' prefix, and is also referred to as a g-file.

If a directory name is used in place of the s.filename argument, the delta command applies to all s. files in that directory. Unreadable s. files produce an error; processing continues with the next file (if any). The use of ‘−’ as the s.filename argument indicates that the names of files are to be read from the standard input, one s. file per line (requires -y, and in some cases, -m).

delta may issue prompts on the standard output depending upon the options specified and the flags that are set in the s. file (see `sccs-admin(1)`, and the -m and -y options below, for details).

The SID of the delta is not echoed to stdout.

OPTIONS

The following options are supported:

- **-d**
  Use command `diff(1)` instead of `bdiff(1)`. Returns exit status 2 if s.filename argument is not specified.

- **-n**
  Retain the edited g-file, which is normally removed at the completion of processing.

- **-p**
  Display line-by-line differences (in `diff(1)` format) on the standard output.

- **-s**
  Silent. Do not display warning or confirmation messages. Do not suppress error messages (which are written to standard error).

- **-g sid-list | -g sid-list**
  Specify a list of deltas to omit when the file is accessed at the SCCS version ID (SID) created by this delta. sid-list is a comma-separated list of SIDs. To specify a range of deltas, use a ‘−’ separator instead of a comma, between two SIDs in the list.

- **-m mr-list | -m mr-list**
  If the SCCS file has the v flag set (see `sccs-admin(1)`), you must supply one or more Modification Request (MR) numbers...
for the new delta. When specifying more than one MR number on the command line, mr-list takes the form of a quoted, space-separated list. If -m is not used and the standard input is a terminal, delta prompts with MRs? for the list (before issuing the comments? prompt). If the v flag in the s.file has a value, it is taken to be the name of a program to validate the MR numbers. If that validation program returns a non-zero exit status, delta terminates without checking in the changes.

- r sid | - rsid

When two or more versions are checked out, specify the version to check in. This SID value can be either the SID specified on the get command line, or the SID of the new version to be checked in as reported by get. A diagnostic results if the specified SID is ambiguous, or if one is required but not supplied.

- y[comment]

Supply a comment for the delta table (version log). A null comment is accepted, and produces an empty commentary in the log. If - y is not specified and the standard input is a terminal, delta prompts with 'comments?'. An unescaped NEWLINE terminates the comment.

EXIT STATUS

The following exit values are returned:

0 Successful completion.
1 An error occurred and the -d option had not been specified.
2 An error occurred, the -d option had been specified, and the s.filename argument was not specified.

FILES
d.file temporary file of differences
p.file lock file for a checked-out version
g.file temporary file
s.file SCCS history file
x.file temporary copy of the s.file
z.file temporary file

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:
sccs-delta(1)

**SEE ALSO**
bdiff(1), diff(1), sccs-admin(1), sccs-cdc(1), sccs-get(1), sccs-help(1),
sccs-prs(1), sccs-prt(1), sccs-rmdel(1), sccs-sccsdiff(1), sccs-unget(1),
sccs(1), what(1), sccsfile(4), attributes(5), XPG4(5)

**WARNINGS**
Lines beginning with an ASCII SOH character (binary 001) cannot be placed in the
SCCS file unless the SOH is escaped. This character has special meaning to SCCS (see
sccsfile(4)) and produces an error.

---

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsprot</td>
</tr>
<tr>
<td>Availability</td>
<td>SUNWxcu4t</td>
</tr>
</tbody>
</table>

**DIAGNOSTICS**
Use the SCCS help command for explanations (see sccs-help(1)).
NAME  sccs-get, get – retrieve a version of an SCCS file

SYNOPSIS  
/usr/ccs/bin/get [-begkmnpst] [-l [p]] [-asequence] [-c date-time | -cdate-time] [-Gg-file] [-i sid-list] [-isid-list] [-x sid-list] [-xsid-list] s.filename...

/usr/xpg4/bin/get [-begkmnpst] [-l [p]] [-asequence] [-c date-time | -cdate-time] [-Gg-file] [-i sid-list] [-isid-list] [-x sid-list] [-xsid-list] s.filename...

DESCRIPTION  The get utility retrieves a working copy from the SCCS history file, according to the specified options.

For each s.filename argument, get displays the SCCS delta ID (SID) and number of lines retrieved.

If a directory name is used in place of the s.filename argument, the get command applies to all s.files in that directory. Unreadable s.files produce an error; processing continues with the next file (if any). The use of ‘−’ as the s.filename argument indicates that the names of files are to be read from the standard input, one s.file per line.

The retrieved file normally has the same filename base as the s.file, less the prefix, and is referred to as the g-file.

For each file processed, get responds (on the standard output) with the SID being accessed, and with the number of lines retrieved from the s.file.

OPTIONS  The following options are supported:

- b  Create a new branch. Used with the -e option to indicate that the new delta should have an SID in a new branch. Instead of incrementing the level for version to be checked in, get indicates in the p.file that the delta to be checked in should either initialize a new branch and sequence (if there is no existing branch at the current level), or increment the branch component of the SID. If the b flag is not set in the s.file, this option is ignored.

- e  Retrieve a version for editing. With this option, get places a lock on the s.file, so that no one else can check in changes to the version you have checked out. If the j flag is set in the s.file, the lock is advisory: get issues a warning message. Concurrent use of ‘get -e’ for different SIDs is allowed; however, get will not check out a version of the file if a writable version is present in the directory. All SCCS file protections stored in the s.file, including the release ceiling, floor, and authorized user list, are honored by ‘get -e’.
sccs-get(1)

-\(g\)  Get the SCCS version ID, without retrieving the version itself. Used to verify the existence of a particular SID.

-\(k\)  Suppress expansion of ID keywords. -\(k\) is implied by the -\(e\).

-\(m\)  Precede each retrieved line with the SID of the delta in which it was added to the file. The SID is separated from the line with a TAB.

-\(n\)  Precede each line with the \(\%M\%\) ID keyword and a TAB. When both the -\(m\) and -\(n\) options are used, the ID keyword precedes the SID, and the line of text.

-\(p\)  Write the text of the retrieved version to the standard output. All messages that normally go to the standard output are written to the standard error instead.

-\(s\)  Suppress all output normally written on the standard output. However, fatal error messages (which always go to the standard error) remain unaffected.

-\(t\)  Retrieve the most recently created (top) delta in a given release (for example: -\(r1\)).

-\(l\)  Retrieve a summary of the delta table (version log) and write it to a listing file, with the ‘l.’ prefix (called ‘l.file’). When -\(lp\) is used, write the summary onto the standard output.

-asequence  Retrieve the version corresponding to the indicated delta sequence number. This option is used primarily by the SCCS comb command (see sccs-comb(1)); for users, -\(r\) is an easier way to specify a version. -\(a\) supersedes -\(r\) when both are used.

-c date-time | -cdate-time  Retrieve the latest version checked in prior to the date and time indicated by the date-time argument. date-time takes the form:

\[yy[mm][dd][hh][mm][ss]]\]

Units omitted from the indicated date and time default to their maximum possible values; that is -\(c7502\) is equivalent to -\(c750228235959\). Values of \(yy\) in the range 69–99 refer to the twentieth century. Values in the range 00–68 refer to the twenty-first century. Any number of non-numeric characters may separate the various 2 digit components. If white-space characters occur, the date-time specification must be quoted.

-\(Gnewname\)  Use newname as the name of the retrieved version.
**sccs-get(1)**

Specify a list of deltas to include in the retrieved version. The included deltas are noted in the standard output message. `sid-list` is a comma-separated list of SIDs. To specify a range of deltas, use a ‘−’ separator instead of a comma, between two SIDs in the list.

```
-isid-list | -isd-list
```

Retrieve the version corresponding to the indicated SID.

```
-r [sid]
```

The SID for a given delta is a number, in Dewey decimal format, composed of two or four fields: the `release` and `level` fields, and for branch deltas, the `branch` and `sequence` fields. For instance, if 1.2 is the SID, 1 is the release, and 2 is the level number. If 1.2.3.4 is the SID, 3 is the branch and 4 is the sequence number.

You need not specify the entire SID to retrieve a version with `get`. When you omit `-r` altogether, or when you omit both release and level, `get` normally retrieves the highest release and level. If the `-d` flag is set to an SID in the `s` file and you omit the SID, `get` retrieves the default version indicated by that flag.

When you specify a release but omit the level, `get` retrieves the highest level in that release. If that release does not exist, `get` retrieves highest level from the next-highest existing release.

Similarly with branches, if you specify a release, level and branch, `get` retrieves the highest sequence in that branch.

```
/usr/ccs/bin/get
```

Same as for `/usr/ccs/bin/get` except that SID is mandatory.

```
-r sid | -rsid
```

Exclude the indicated deltas from the retrieved version. The excluded deltas are noted in the standard output message. `sid-list` is a comma-separated list of SIDs. To specify a range of deltas, use a ‘−’ separator instead of a comma, between two SIDs in the list.

```
-xsd-list | -xsid-list
```

The output format for `/usr/ccs/bin/get` is as follows:

```%
%s
%d

<sid>, <number of lines>
```

The output format for `/usr/xpg4/bin/get` is as follows:

```%
%s
%d

<sid>, <number of lines>
```
In the absence of `-e` or `-k`, `get` expands the following ID keywords by replacing them with the indicated values in the text of the retrieved source.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Value</th>
</tr>
</thead>
</table>
| `%A%`  | Shorthand notation for an ID line with data for `what(1)`:
|        | `%Z%%Y% %M% %I%%T%` |
| `%B%`  | SID branch component |
| `%C%`  | Current line number. Intended for identifying messages output by the program such as “this shouldn’t have happened” type errors. It is not intended to be used on every line to provide sequence numbers. |
| `%D%`  | Current date: `yy/mm/dd` |
| `%E%`  | Date newest applied delta was created: `yy/mm/dd` |
| `%F%`  | SCCS `.file` name |
| `%G%`  | Date newest applied delta was created: `mm/dd/yy` |
| `%H%`  | Current date: `mm/dd/yy` |
| `%I%`  | SID of the retrieved version: `%R%.%L%.%B%.%S%` |
| `%L%`  | SID level component |
| `%M%`  | Module name: either the value of the `m` flag in the `.file` (see `sccs-admin(1)`), or the name of the `.file` less the prefix |
| `%P%`  | Fully qualified `.file` name |
| `%Q%`  | Value of the `q` flag in the `.file` |
| `%R%`  | SID Release component |
| `%S%`  | SID Sequence component |
| `%T%`  | Current time: `hh:mm:ss` |
| `%U%`  | Time the newest applied delta was created: `hh:mm:ss` |
| `%V%`  | Shorthand notation for an ID line with data for `what(1)`:
|        | `%Z%%M% %I%` |
| `%Y%`  | Module type: value of the `t` flag in the `.file` |
| `%Z%`  | 4-character string: ‘@(#)’, recognized by `what` |

The table below explains how the SCCS identification string is determined for retrieving and creating deltas.
### Determination of SCCS Identification String

<table>
<thead>
<tr>
<th>SID (1) Specified</th>
<th>-b Option Used (2)</th>
<th>Other Conditions</th>
<th>SID Retrieved</th>
<th>SID of Delta to be Created</th>
</tr>
</thead>
<tbody>
<tr>
<td>none (3)</td>
<td>no</td>
<td>R defaults to mR</td>
<td>mR.mL</td>
<td>mR.(mL+1)</td>
</tr>
<tr>
<td>none (3)</td>
<td>yes</td>
<td>R defaults to mR</td>
<td>mR.mL</td>
<td>mR.mL.(mB+1).1</td>
</tr>
<tr>
<td>R</td>
<td>no</td>
<td>R &gt; mR</td>
<td>mR.mL</td>
<td>R.1 (4)</td>
</tr>
<tr>
<td>R</td>
<td>yes</td>
<td>R &gt; mR</td>
<td>mR.mL</td>
<td>mR.mL.(mB+1).1</td>
</tr>
<tr>
<td>R</td>
<td>yes</td>
<td>R &lt; mR and R does not exist</td>
<td>hR.mL (5)</td>
<td>hR.mL.(mB+1).1</td>
</tr>
<tr>
<td>R</td>
<td>–</td>
<td>Trunk succ. (6) in release &gt; R and R exists</td>
<td>R.mL</td>
<td>R.mL.(mB+1).1</td>
</tr>
<tr>
<td>R.L</td>
<td>no</td>
<td>No trunk succ.</td>
<td>R.L</td>
<td>R.(L+1)</td>
</tr>
<tr>
<td>R.L</td>
<td>yes</td>
<td>No trunk succ.</td>
<td>R.L</td>
<td>R.L.(mB+1).1</td>
</tr>
<tr>
<td>R.L</td>
<td>–</td>
<td>Trunk succ. in release ≥ R</td>
<td>R.L</td>
<td>R.L.(mB+1).1</td>
</tr>
<tr>
<td>R.L.B</td>
<td>no</td>
<td>No branch succ.</td>
<td>R.L.B.mS</td>
<td>R.L.B.(mS+1)</td>
</tr>
<tr>
<td>R.L.B</td>
<td>yes</td>
<td>No branch succ.</td>
<td>R.L.B.mS</td>
<td>R.L.(mB+1).1</td>
</tr>
<tr>
<td>R.L.B.S</td>
<td>no</td>
<td>No branch succ.</td>
<td>R.L.B.S</td>
<td>R.L.B.(S+1)</td>
</tr>
<tr>
<td>R.L.B.S</td>
<td>yes</td>
<td>No branch succ.</td>
<td>R.L.B.S</td>
<td>R.L.(mB+1).1</td>
</tr>
<tr>
<td>R.L.B.S</td>
<td>–</td>
<td>Branch succ.</td>
<td>R.L.B.S</td>
<td>R.L.(mB+1).1</td>
</tr>
</tbody>
</table>

1. ‘R’, ‘L’, ‘B’, and ‘S’ are the ‘release’, ‘level’, ‘branch’, and ‘sequence’ components of the SID, respectively; ‘m’ means ‘maximum’. Thus, for example, ‘R.mL’ means ‘the maximum level number within release R’; ‘R.L.(mB+1).1’ means ‘the first sequence number on the new branch (that is, maximum branch number plus one) of level L within release R’. Note: if the SID specified is of the form ‘R.L’, ‘R.L.B’, or ‘R.L.B.S’, each of the specified components must exist.

2. The -b option is effective only if the b flag is present in the file. An entry of ‘-’ means ‘irrelevant’.

---

**User Commands**
sccs-get(1)

(3) This case applies if the ñ (default SID) flag is *not* present in the file. If the ñ flag *is* present in the file, the SID obtained from the ñ flag is interpreted as if it had been specified on the command line. Thus, one of the other cases in this table applies.

(4) Forces creation of the first delta in a new release.

(5) ‘hR’ is the highest existing release that is lower than the specified, nonexistent, release R.

(6) Successor.

FILES

g-file version retrieved by get
l .file file containing extracted delta table info
p .file permissions (lock) file
z .file temporary copy of s .file

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

<table>
<thead>
<tr>
<th>/usr/ccs/bin/get</th>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsprot</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>/usr/xpg4/bin/get</th>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWxcu4t</td>
<td></td>
</tr>
</tbody>
</table>


DIAGNOSTICS Use the SCCS help command for explanations (see sccs-help(1)).

BUGS If the effective user has write permission (either explicitly or implicitly) in the directory containing the SCCS files, but the real user does not, only one file may be named when using -e.
NAME | sccs-help, help – ask for help regarding SCCS error or warning messages
SYNOPSIS | /usr/ccs/bin/help [argument…]
DESCRIPTION | The help utility retrieves information to further explain errors messages and warnings from SCCS commands. It also provides some information about SCCS command usage. If no arguments are given, help prompts for one.

An argument may be a message number (which normally appears in parentheses following each SCCS error or warning message), or an SCCS command name. help responds with an explanation of the message or a usage line for the command.

When all else fails, try ‘/usr/ccs/bin/help stuck’.

FILES | /usr/lib/help directory containing files of message text
ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsprot</td>
</tr>
</tbody>
</table>

**NAME**  
sccs-prs, prs – display selected portions of an SCCS history

**SYNOPSIS**  
/usr/ccs/bin/prs [-ael] [-c date-time] [-d dataspec] [-r sid] s.filename...

**DESCRIPTION**  
The prs utility displays part or all of the SCCS file (see sccsfile(4)) in a user supplied format.

If a directory utility name is used in place of the s.filename argument, the prs command applies to all s.files in that directory. Unreadable s.files produce an error; processing continues with the next file (if any). The use of ‘-’ as the s.filename argument indicates that the names of files are to be read from the standard input, one s.file per line.

**OPTIONS**  
In the absence of options, prs displays the delta table (version log). In the absence of -d, or -l, prs displays the entry for each delta indicated by the other options.

- **-a**  
  Includes all deltas, including those marked as removed (see sccs-rmdel(1)).

- **-e**  
  Requests information for all deltas created earlier than, and including, the delta indicated with -r or -c.

- **-l**  
  Requests information for all deltas created later than, and including, the delta indicated with -r or -c.

- **-c date-time**  
  Either options -e or -l must be used with this option. 
  -c date-time displays information on the deltas checked in either prior to and including the date and time indicated by the date-time argument (option -e); or later than and including the date and time indicated (option -l). date-time takes the form:

  yy[mm][dd][hh][mm][ss]]

  Units omitted from the indicated date and time default to their maximum possible values; that is -c7502 is equivalent to -c750228235959. Any number of non-numeric characters may separate the various 2 digit components. If white-space characters occur, the date-time specification must be quoted. Values of yy in the range 69–99 refer to the twentieth century. Values in the range of 00–68 refer to the twenty-first century.

- **-d dataspec**  
  Produce a report according to the indicated data specification. dataspec consists of a (quoted) text string that includes embedded data keywords of the form:'key:' (see Data Keywords, below). prs expands these keywords in the output it produces. To specify a TAB character in the output, use \t; to specify a NEWLINE in the output, use \n.

- **-r sid**  
  Specifies the SCCS delta ID (SID) of the delta for which information is desired. If no SID is specified, the most recently created delta is used.
Data Keywords

Data keywords specify which parts of an SCCS file are to be retrieved. All parts of an SCCS file (see `sccsfile(4)`) have an associated data keyword. A data keyword may appear any number of times in a data specification argument to `-d`. These data keywords are listed in the table below:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Data Item</th>
<th>File Section*</th>
<th>Value</th>
<th>Format**</th>
</tr>
</thead>
<tbody>
<tr>
<td>:B:</td>
<td>branch number</td>
<td>D</td>
<td>mmm</td>
<td>S</td>
</tr>
<tr>
<td>:BD:</td>
<td>body</td>
<td>B</td>
<td>text</td>
<td>M</td>
</tr>
<tr>
<td>:BF:</td>
<td>branch flag</td>
<td>F</td>
<td>yes or no</td>
<td>S</td>
</tr>
<tr>
<td>:CB:</td>
<td>ceiling boundary</td>
<td>F</td>
<td>:R:</td>
<td>S</td>
</tr>
<tr>
<td>:C:</td>
<td>comments for delta</td>
<td>D</td>
<td>text</td>
<td>M</td>
</tr>
<tr>
<td>:D:</td>
<td>date delta created</td>
<td>D</td>
<td>:Dy:/:Dm:/:Dd:</td>
<td>S</td>
</tr>
<tr>
<td>:Dd:</td>
<td>day delta created</td>
<td>D</td>
<td>m</td>
<td>S</td>
</tr>
<tr>
<td>:Dg:</td>
<td>deltas ignored (seq #)</td>
<td>D</td>
<td>:DS: :DS: ...</td>
<td>S</td>
</tr>
<tr>
<td>:DI:</td>
<td>seq-no. of deltas included, excluded, ignored</td>
<td>D</td>
<td>:Dn:/:Dx:/:Dg:</td>
<td>S</td>
</tr>
<tr>
<td>:DL:</td>
<td>delta line statistics</td>
<td>D</td>
<td>:Li:/:Ld:/:Lu:</td>
<td>S</td>
</tr>
<tr>
<td>:Dm:</td>
<td>month delta created</td>
<td>D</td>
<td>m</td>
<td>S</td>
</tr>
<tr>
<td>:Dn:</td>
<td>deltas included (seq #)</td>
<td>D</td>
<td>:DS: :DS: ...</td>
<td>S</td>
</tr>
<tr>
<td>:DP:</td>
<td>predecessor delta seq-no.</td>
<td>D</td>
<td>mmm</td>
<td>S</td>
</tr>
<tr>
<td>:DS:</td>
<td>default SID</td>
<td>F</td>
<td>:I:</td>
<td>S</td>
</tr>
<tr>
<td>:DS:</td>
<td>delta sequence number</td>
<td>D</td>
<td>mmm</td>
<td>S</td>
</tr>
<tr>
<td>:DT:</td>
<td>delta type</td>
<td>D</td>
<td>D or R</td>
<td>S</td>
</tr>
<tr>
<td>:Dx:</td>
<td>deltas excluded (seq #)</td>
<td>D</td>
<td>:DS: ...</td>
<td>S</td>
</tr>
<tr>
<td>:Dy:</td>
<td>year delta created</td>
<td>D</td>
<td>m</td>
<td>S</td>
</tr>
<tr>
<td>:F:</td>
<td>s.file name</td>
<td>N/A</td>
<td>text</td>
<td>S</td>
</tr>
<tr>
<td>:FB:</td>
<td>floor boundary</td>
<td>F</td>
<td>:R:</td>
<td>S</td>
</tr>
<tr>
<td>:FD:</td>
<td>file descriptive text</td>
<td>C</td>
<td>text</td>
<td>M</td>
</tr>
<tr>
<td>:FL:</td>
<td>flag list</td>
<td>F</td>
<td>text</td>
<td>M</td>
</tr>
</tbody>
</table>

`sccs-prs(1)`
### sccs-prs(1)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Data Item</th>
<th>File Section*</th>
<th>Value</th>
<th>Format**</th>
</tr>
</thead>
<tbody>
<tr>
<td>:GB:</td>
<td>gotten body</td>
<td>B</td>
<td>text</td>
<td>M</td>
</tr>
<tr>
<td>:I:</td>
<td>SCCS delta ID (SID)</td>
<td>D</td>
<td>:R::L::B::S:</td>
<td>S</td>
</tr>
<tr>
<td>:J:</td>
<td>joint edit flag</td>
<td>F</td>
<td>yes or no</td>
<td>S</td>
</tr>
<tr>
<td>:KF:</td>
<td>keyword error/warning flag</td>
<td>F</td>
<td>yes or no</td>
<td>S</td>
</tr>
<tr>
<td>:L:</td>
<td>level number</td>
<td>D</td>
<td>mmm</td>
<td>S</td>
</tr>
<tr>
<td>:Ld:</td>
<td>lines deleted by delta</td>
<td>D</td>
<td>mmmm</td>
<td>S</td>
</tr>
<tr>
<td>:Li:</td>
<td>lines inserted by delta</td>
<td>D</td>
<td>mmmm</td>
<td>S</td>
</tr>
<tr>
<td>:LK:</td>
<td>locked releases</td>
<td>F</td>
<td>:R:...</td>
<td>S</td>
</tr>
<tr>
<td>:Lu:</td>
<td>lines unchanged by delta</td>
<td>D</td>
<td>mmmm</td>
<td>S</td>
</tr>
<tr>
<td>:M:</td>
<td>module name</td>
<td>F</td>
<td>text</td>
<td>S</td>
</tr>
<tr>
<td>:MF:</td>
<td>MR validation flag</td>
<td>F</td>
<td>yes or no</td>
<td>S</td>
</tr>
<tr>
<td>:MP:</td>
<td>MR validation program</td>
<td>F</td>
<td>text</td>
<td>S</td>
</tr>
<tr>
<td>:MR:</td>
<td>MR numbers for delta</td>
<td>D</td>
<td>text</td>
<td>M</td>
</tr>
<tr>
<td>:ND:</td>
<td>null delta flag</td>
<td>F</td>
<td>yes or no</td>
<td>S</td>
</tr>
<tr>
<td>:Q:</td>
<td>user defined keyword</td>
<td>F</td>
<td>text</td>
<td>S</td>
</tr>
<tr>
<td>:P:</td>
<td>user who created delta</td>
<td>D</td>
<td>username</td>
<td>S</td>
</tr>
<tr>
<td>:PN:</td>
<td>a file's pathname</td>
<td>N/A</td>
<td>text</td>
<td>S</td>
</tr>
<tr>
<td>:R:</td>
<td>release number</td>
<td>D</td>
<td>mmm</td>
<td>S</td>
</tr>
<tr>
<td>:S:</td>
<td>sequence number</td>
<td>D</td>
<td>mmm</td>
<td>S</td>
</tr>
<tr>
<td>:T:</td>
<td>time delta created</td>
<td>D</td>
<td>:Th::Tm::Ts:</td>
<td>S</td>
</tr>
<tr>
<td>:Th:</td>
<td>hour delta created</td>
<td>D</td>
<td>mm</td>
<td>S</td>
</tr>
<tr>
<td>:Tm:</td>
<td>minutes delta created</td>
<td>D</td>
<td>mm</td>
<td>S</td>
</tr>
<tr>
<td>:Ts:</td>
<td>seconds delta created</td>
<td>D</td>
<td>mm</td>
<td>S</td>
</tr>
<tr>
<td>:UN:</td>
<td>user names</td>
<td>U</td>
<td>text</td>
<td>M</td>
</tr>
<tr>
<td>:W:</td>
<td>a form of what string</td>
<td>N/A</td>
<td>:Z::M:\t::I:</td>
<td>S</td>
</tr>
<tr>
<td>:Y:</td>
<td>module type flag</td>
<td>F</td>
<td>text</td>
<td>S</td>
</tr>
<tr>
<td>:Z:</td>
<td>what string delimiter</td>
<td>N/A</td>
<td>@(#)</td>
<td>S</td>
</tr>
</tbody>
</table>

*B = body, D = delta table, F = flags, U = user names

**S = simple format, M = multi-line format
EXAMPLE 1 Displaying delta entries

The following command:
```
example% /usr/ccs/bin/prs -e -d*:I:\t:P:* program.c
```
produces:
```
1.6 username
1.5 username...
```

FILES /tmp/pr????? temporary file

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsprot</td>
</tr>
</tbody>
</table>

SEE ALSO sccs(1), sccs-cdc(1), sccs-delta(1), sccs-get(1), sccs-help(1), sccs-prt(1), sccs-sact(1), sccs-sccsdiff(1), what(1), sccsfile(4), attributes(5)

DIAGNOSTICS Use the SCCS help command for explanations (see sccs-help(1)).
NAME  sccs-prt, prt – display delta table information from an SCCS file

SYNOPSIS  

/usr/ccs/bin/prt [-abdefistu] [-c date-time] [-r date-time] [-y sid] s.filename...

DESCRIPTION  prt prints selected portions of an SCCS file. By default, it prints the delta table (version log).

If a directory name is used in place of the s.filename argument, the prt command applies to all s.files in that directory. Unreadable s.files produce an error; processing continues with the next file (if any). The use of ‘−’ as the s.filename argument indicates that the names of files are to be read from the standard input, one s.file per line.

OPTIONS  If any option other than −y, −c, or −r is supplied, the name of each file being processed (preceded by one NEWLINE and followed by two NEWLINE characters) appears above its contents.

If none of the −u, −f, −t, or −b options are used, −d is assumed. −a, −i are mutually exclusive, as are −c and −r.

- a  Display log entries for all deltas, including those marked as removed.

- b  Print the body of the s.file.

- d  Print delta table entries. This is the default.

- e  Everything. This option implies −d, −i, −u, −f, and −t.

- f  Print the flags of each named s.file.

- i  Print the serial numbers of included, excluded, and ignored deltas.

- s  Print only the first line of the delta table entries; that is, only up to the statistics.

- t  Print the descriptive text contained in the s.file.

- u  Print the user-names and/or numerical group IDs of users allowed to make deltas.

- cdate-time  Exclude delta table entries that are specified cutoff date and time. Each entry is printed as a single line, preceded by the name of the SCCS file. This format (also produced by −r, and −y) makes it easy to sort multiple delta tables in chronological order. When both −y and −c, or −y and −r are supplied, prt stops printing when the first of the two conditions is met.

- rdate-time  Exclude delta table entries that are newer than the specified cutoff date and time.

- ysid  Exclude delta table entries made prior to the SID specified. If no delta in the table has the specified SID, the entire table is printed. If no SID is specified, the most recent delta is printed.
Output Format
The following format is used to print those portions of the $s$. file that are specified by
the various options.

- NEWLINE
- Type of delta (D or R)
- SPACE
- SCCS delta ID (SID)
- TAB
- Date and time of creation in the form: $yy/mm/dd $hh/mm/ss
- SPACE
- Username the delta’s creator
- TAB
- Serial number of the delta
- SPACE
- Predecessor delta’s serial number
- TAB
- Line-by-line change statistics in the form: inserted/deleted/unchanged
- NEWLINE
- List of included deltas, followed by a NEWLINE (only if there were any such
deltas and the -i options was used)
- List of excluded deltas, followed by a NEWLINE (only if there were any such
deltas and the -i options was used)
- List of ignored deltas, followed by a NEWLINE (only if there were any such deltas
and the -i options was used)
- List of modification requests (MRs), followed by a NEWLINE (only if any MR
numbers were supplied).
- Lines of the delta commentary (if any), followed by a NEWLINE.

EXAMPLES
EXAMPLE 1 Examples of prt.

The following command:

example% /usr/ccs/bin/prt -y program.c

produces a one-line display of the delta table entry for the most recent version:

s.program.c: D 1.6 88/07/06 21:39:39 username 5 4
00159/00080/00636...

ATTRIBUTES
See attributes(5) for descriptions of the following attributes:
Use the SCCS help command for explanations (see `sccs-help(1)`).

**SEE ALSO**

`sccs(1), sccs-cdc(1), sccs-delta(1), sccs-get(1), sccs-help(1), sccs-prs(1),
sccs-sact(1), sccs-sccsdiff(1), what(1), sccsfile(4), attributes(5)`

**DIAGNOSTICS**

Use the SCCS help command for explanations (see `sccs-help(1)`).
NAME sccs-rmdel, rmdel – remove a delta from an SCCS file

SYNOPSIS /usr/ccs/bin/rmdel -rsid s.filename...

DESCRIPTION rmdel removes the delta specified by the SCCS delta ID (SID) supplied with -r. The delta to be removed must be the most recent (leaf) delta in its branch. In addition, the SID must not be that of a version checked out for editing: it must not appear in any entry of the version lock file (p.file).

If you created the delta, or, if you own the file and directory and have write permission, you can remove it with rmdel.

If a directory name is used in place of the s.filename argument, the rmdel command applies to all s.files in that directory. Unreadable s.files produce an error; processing continues with the next file (if any). The use of ‘−’ as the s.filename argument indicates that the names of files are to be read from the standard input, one s.file per line.

OPTIONS The following option is supported:

- rsid Remove the version corresponding to the indicated SID (delta).

FILES p.file permissions file
s.file history file
z.file temporary copy of the s.file

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsprot</td>
</tr>
</tbody>
</table>

SEE ALSO sccs(1), sccs-admin(1), sccs-cdc(1), sccs-comb(1), sccs-delta(1),
sccs-help(1), sccs-prs(1), sccs-prt(1), sccs-sccsdiff(1), sccs-unget(1),
what(1), sccsfile(4), attributes(5)

DIAGNOSTICS Use the SCCS help command for explanations (see sccs-help(1)).
NAME  sccs-sact, sact – show editing activity status of an SCCS file

SYNOPSIS  /usr/ccs/bin/sact s.filename...

DESCRIPTION  sact informs the user of any SCCS files that are checked out for editing.

The output for each named file consists of five fields separated by SPACE characters.

- SID of a delta that currently exists in the SCCS file, to which changes will be made
to make the new delta
- SID for the new delta to be created
- Username of the person who has the file checked out for editing.
- Date that the version was checked out.
- Time that the version was checked out.

If a directory name is used in place of the s.filename argument, the sact command applies to all s.files in that directory. Unreadable s.files produce an error; processing continues with the next file (if any). The use of ‘−’ as the s.filename argument indicates that the names of files are to be read from the standard input, one s.file per line.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNW'sprot</td>
</tr>
</tbody>
</table>

SEE ALSO  sccs(1), sccs-delta(1), sccs-get(1), sccs-help(1), sccs-prs(1), sccs-prt(1),
what(1), sccsfile(4), attributes(5)

DIAGNOSTICS  Use the SCCS help command for explanations (see sccs-help(1)).
NAME
sccs-sccsdiff, sccsdiff – compare two versions of an SCCS file

SYNOPSIS
/usr/ccs/bin/sccsdiff [-p] -rsid -rsid [diff-options] s.filename

DESCRIPTION
sccsdiff compares two versions of an SCCS file and displays the differences between the two versions. Any number of SCCS files may be specified. The options specified apply to all named s.files.

OPTIONS
The following options are supported:

-`p` Pipe output for each file through pr(1).
-`rsid` Specify a version corresponding to the indicated SCCS delta ID (SID) for comparison. Versions are passed to diff(1) in the order given.

diff-options Pass options to diff(1), including: -b, -c, -e, -f, -h, -u, -C number, -U number, and -D string.

FILES
/tmp/get????? temporary files

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsprot</td>
</tr>
</tbody>
</table>

SEE ALSO
diff(1), sccs(1), sccs-delta(1), sccs-get(1), sccs-help(1), sccs-prs(1), sccs-prt(1), what(1), sccsfile(4), attributes(5)

DIAGNOSTICS
filename: No differences If the two versions are the same.

Use the SCCS help command for explanations of other messages. See sccs-help(1).
NAME | sccs-unget, unget – undo a previous get of an SCCS file
SYNOPSIS | /usr/ccs/bin/unget [-ns] [-rsid] s.filename...
DESCRIPTION | unget undoes the effect of a ‘get -e’ done prior to the creation of the pending delta.

If a directory name is used in place of the s.filename argument, the unget command applies to all s.files in that directory. Unreadable s.files produce an error; processing continues with the next file (if any). The use of ‘−’ as the s.filename argument indicates that the names of files are to be read from the standard input, one s.file per line.

OPTIONS | - n Retain the retrieved version, which is otherwise removed.
- s Suppress display of the SCCS delta ID (SID).
- rsid When multiple versions are checked out, specify which pending delta to abort. A diagnostic results if the specified SID is ambiguous, or if it is necessary but omitted from the command line.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsprot</td>
</tr>
</tbody>
</table>

SEE ALSO | sccs(1), sccs-delta(1), sccs-get(1), sccs-help(1), sccs-prs(1), sccs-prt(1), sccs-rmdel(1), sccs-sact(1), sccs-sccsdiff(1), what(1), sccsfile(4), attributes(5)

DIAGNOSTICS | Use the SCCS help command for explanations (see sccs-help(1)).
sccs-val(1)

NAME
sccs-val, val – validate an SCCS file

SYNOPSIS
/usr/ccs/bin/val -
/usr/ccs/bin/val [-s] [-m name] [-r sid] [-y type] s.filename...

DESCRIPTION
val determines if the specified s. files meet the characteristics specified by the
indicated arguments. val can process up to 50 files on a single command line.
val has a special argument, ‘−’, which reads the standard input until the end-of-file
condition is detected. Each line read is independently processed as if it were a
command line argument list.

val generates diagnostic messages on the standard output for each command line and
file processed and also returns a single 8–bit code upon exit as described below.

The 8-bit code returned by val is a disjunction of the possible errors, that is, it can be
interpreted as a bit string where (moving from left to right) the bits set are interpreted
as follows:

bit 0 = missing file argument
bit 1 = unknown or duplicate option
bit 2 = corrupted s.file
bit 3 = can not open file or file not in s.file format
bit 4 = the SCCS delta ID (SID) is invalid or ambiguous
bit 5 = the SID does not exist
bit 6 = mismatch between −m argument
bit 7 = mismatch between sccs-val.1 −m argument

val can process two or more files on a given command line, and in turn can process
multiple command lines (when reading the standard input). In these cases, an
aggregate code is returned which is the logical OR of the codes generated for each
command line and file processed.

OPTIONS
The following options are supported:

-s Silent. Suppress the normal error or warning messages.

-m name Compare name with the sccs-val.1 ID keyword in the s.file.

-r sid Check to see if the indicated SID is ambiguous, invalid, or absent from the
s.file.

-y type Compare type with the ID keyword.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsprot</td>
</tr>
</tbody>
</table>
### SEE ALSO

| sccs(1), sccs-admin(1), sccs-delta(1), sccs-get(1), sccs-help(1), what(1), sccsfile(4), attributes(5) |

### DIAGNOSTICS

Use the SCCS help command for explanations (see sccs-help(1)).
scp(1)

NAME
scp – secure copy (remote file copy program)

SYNOPSIS
scp [-pqrvC46] [-S program] [-P port] [-c cipher] [-i identity_file] [-o option] [ [user1@]host1:]file1 [ [user2@]host2:]file2 [...]

DESCRIPTION
The scp utility copies files between hosts on a network. It uses ssh(1) for data transfer, and uses the same authentication and provides the same security as ssh(1). Unlike rcp(1), scp will ask for passwords or passphrases if they are needed for authentication.

Any file name may contain a host and user specification to indicate that the file is to be copied to/from that host. Copies between two remote hosts are permitted.

OPTIONS
The following options are supported:

-4 Forces scp to use IPv4 addresses only.
-6 Forces scp to use IPv6 addresses only.
-B Selects batch mode. (Prevents asking for passwords or passphrases.)
-c cipher Selects the cipher to use for encrypting the data transfer. This option is directly passed to ssh(1).
-C Compression enable. Passes the -C flag to ssh(1) to enable compression.
-i identity_file Selects the file from which the identity (private key) for RSA authentication is read. This option is directly passed to ssh(1).
-o option The given option is directly passed to ssh(1).
-p Preserves modification times, access times, and modes from the original file.
-P port Specifies the port to connect to on the remote host. Notice that this option is written with a capital 'P', because -p is already reserved for preserving the times and modes of the file in rcp(1).
-q Disables the progress meter.
-r Recursively copies entire directories.
-S program Specifies the name of the program to use for the encrypted connection. The program must understand ssh(1) options.
-v Verbose mode. Causes scp and ssh(1) to print debugging messages about their progress. This is helpful in debugging connection, authentication, and configuration problems.

OPERANDS
The following operands are supported:

host1, host2,... The name(s) of the host from or to which the file is to be copied.
file1, file2,... The file(s) to be copied.
scp(1)

EXIT STATUS
The following exit values are returned:
0 Successful completion.
1 An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsshu</td>
</tr>
</tbody>
</table>

SEE ALSO
rcp(1), ssh(1), ssh-add(1), ssh-agent(1), ssh-keygen(1), sshd(1M), attributes(5)

To view license terms, attribution, and copyright for OpenSSH, the default path is /var/sadm/pkg/SUNWsshdr/install/copyright. If the Solaris operating environment has been installed anywhere other than the default, modify the given path to access the file at the installed location.

AUTHORS
scp is based on the rcp(1) program in the BSD source code from the Regents of the University of California. The authors are Timo Rinne and Tatu Ylonen.
script makes a record of everything printed on your screen. The record is written to filename. If no file name is given, the record is saved in the file typescript.

The script command forks and creates a sub-shell, according to the value of $SHELL, and records the text from this session. The script ends when the forked shell exits or when CTRL-D is typed.

OPTIONS

-a Append the session record to filename, rather than overwrite it.

NOTES

script places everything that appears on the screen in filename, including prompts.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

SEE ALSO

attributes(5)
sdiff(1)

NAME       sdiff – print differences between two files side-by-side

SYNOPSIS   sdiff [-l] [-s] [-o output] [-w n] filename1 filename2

DESCRIPTION sdiff uses the output of the diff command to produce a side-by-side listing of two
files indicating lines that are different. Lines of the two files are printed with a blank
gutter between them if the lines are identical, a < in the gutter if the line appears only
in filename1, a > in the gutter if the line appears only in filename2, and a | for lines that
are different. (See the EXAMPLES section below.)

OPTIONS    -l  Print only the left side of any lines that are identical to
             -s  Do not print identical lines.
             -o output  Use the argument output as the name of a third file that is created
                        as a user-controlled merge of filename1 and filename2. Identical lines
                        of filename1 and filename2 are copied to output. Sets of differences,
                        as produced by diff, are printed; where a set of differences share
                        a common gutter character. After printing each set of differences,
                        sdiff prompts the user with a % and waits for one of the
                        following user-typed commands:
                        l  Append the left column to the output file.
                        r  Append the right column to the output file.
                        s  Turn on silent mode; do not print identical lines.
                        v  Turn off silent mode.
                        e l  Call the editor with the left column.
                        e r  Call the editor with the right column.
                        e b  Call the editor with the concatenation of left and right.
                        e   Call the editor with a zero length file.
                        q   Exit from the program.

On exit from the editor, the resulting file is concatenated to the end
of the output file.

             -w n  Use the argument n as the width of the output line. The default
                    line length is 130 characters.

USAGE       See largefile(5) for the description of the behavior of sdiff when encountering
            files greater than or equal to 2 Gbyte (2^31 bytes).

EXAMPLES   EXAMPLE 1 An example of the sdiff command.

            A sample output of sdiff follows.
            x   y
            a   a
            b   <

1344       man pages section 1: User Commands • Last Revised 20 Dec 1996
EXAMPLE 1 An example of the sdiff command.  

```
c <
d > c
```

ENVIRONMENT VARIABLES

If any of the LC_* variables (LC_CTYPE, LC_MESSAGES, LC_TIME, LC_COLLATE, LC_NUMERIC, and LC_MONETARY) (see environ(5)) are not set in the environment, the operational behavior of sdiff for each corresponding locale category is determined by the value of the LANG environment variable. If LC_ALL is set, its contents are used to override both the LANG and the other LC_* variables. If none of the above variables is set in the environment, the "C" locale determines how sdiff behaves.

LC_CTYPE Determines how sdiff handles characters. When LC_CTYPE is set to a valid value, sdiff can display and handle text and filenames containing valid characters for that locale.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

SEE ALSO
diff(1), ed(1), attributes(5), environ(5), largefile(5)
sed(1)

NAME    sed – stream editor

SYNOPSIS /usr/bin/sed [ -n ] script  [file...]
   /usr/bin/sed [ -n ] [-e script]... [-f script_file]... [file...]
   /usr/xpg4/bin/sed [ -n ] script  [file...]
   /usr/xpg4/bin/sed [ -n ] [-e script]... [-f script_file]... [file...]

DESCRIPTION The sed utility is a stream editor that reads one or more text files, makes editing
changes according to a script of editing commands, and writes the results to standard
output. The script is obtained from either the script operand string, or a combination of
the option-arguments from the -e script and -f script_file options.

The sed utility is a text editor. It cannot edit binary files or files containing ASCII NUL
(\0) characters or very long lines.

OPTIONS The following options are supported:
   -e script  script is an edit command for sed. See USAGE below for more
               information on the format of script. If there is just one -e option
               and no -f options, the flag -e may be omitted.
   -f script_file  Take the script from script_file. script_file consists of editing
                   commands, one per line.
   -n  Suppress the default output.

Multiple -e and -f options may be specified. All commands are added to the script in
the order specified, regardless of their origin.

OPERANDS The following operands are supported:
   file  A path name of a file whose contents will be read and edited. If
          multiple file operands are specified, the named files will be read in
          the order specified and the concatenation will be edited. If no file
          operands are specified, the standard input will be used.
   script  A string to be used as the script of editing commands. The
           application must not present a script that violates the restrictions of
           a text file except that the final character need not be a NEWLINE
           character.

USAGE A script consists of editing commands, one per line, of the following form:

[ address  [ , address ] ] command [ arguments ]

Zero or more blank characters are accepted before the first address and before
command. Any number of semicolons are accepted before the first address.
In normal operation, *sed* cyclically copies a line of input (less its terminating NEWLINE character) into a *pattern space* (unless there is something left after a `D` command), applies in sequence all commands whose *addresses* select that pattern space, and copies the resulting pattern space to the standard output (except under `-n`) and deletes the pattern space. Whenever the pattern space is written to standard output or a named file, *sed* will immediately follow it with a NEWLINE character.

Some of the commands use a *hold space* to save all or part of the *pattern space* for subsequent retrieval. The *pattern* and *hold spaces* will each be able to hold at least 8192 bytes.

**sed Addresses**

An *address* is either empty, a decimal number that counts input lines cumulatively across files, a `$` that addresses the last line of input, or a context address, which consists of a `/regular expression/` as described on the `regexp(5)` manual page.

A command line with no addresses selects every pattern space.

A command line with one address selects each pattern space that matches the address.

A command line with two addresses selects the inclusive range from the first pattern space that matches the first address through the next pattern space that matches the second address. Thereafter the process is repeated, looking again for the first address. (If the second address is a number less than or equal to the line number selected by the first address, only the line corresponding to the first address is selected.)

Typically, address are separated from each other by a comma (,). They may also be separated by a semicolon (;).

**sed Regular Expressions**

*sed* supports the basic regular expressions described on the `regexp(5)` manual page, with the following additions:

\cREc  In a context address, the construction `\cREc`, where `c` is any character other than a backslash or NEWLINE character, is identical to `/RE/`. If the character designated by `c` appears following a backslash, then it is considered to be that literal character, which does not terminate the RE. For example, in the context address `\abc\xdefx`, the second `x` stands for itself, so that the regular expression is `abc\xdef`.

\n  The escape sequence `\n` matches a NEWLINE character embedded in the pattern space. A literal NEWLINE character must not be used in the regular expression of a context address or in the substitute command.

Editing commands can be applied only to non-selected pattern spaces by use of the negation command `!` (described below).

In the following list of functions the maximum number of permissible addresses for each function is indicated.

The `x` and `w` commands take an optional `rfile` (or `wfile`) parameter, separated from the command letter by one or more blank characters.
Multiple commands can be specified by separating them with a semicolon (;) on the same command line.

The text argument consists of one or more lines, all but the last of which end with \ to hide the NEWLINE. Each embedded NEWLINE character in the text must be preceded by a backslash. Other backslashes in text are removed and the following character is treated literally. Backslashes in text are treated like backslashes in the replacement string of an s command, and may be used to protect initial blanks and tabs against the stripping that is done on every script line. The rfile or wfile argument must terminate the command line and must be preceded by exactly one blank. The use of the wfile parameter causes that file to be initially created, if it does not exist, or will replace the contents of an existing file. There can be at most 10 distinct wfile arguments.

Regular expressions match entire strings, not just individual lines, but a NEWLINE character is matched by \n in a sed RE; a NEWLINE character is not allowed in an RE. Also note that \n cannot be used to match a NEWLINE character at the end of an input line; NEWLINE characters appear in the pattern space as a result of the N editing command.

Two of the commands take a command-list, which is a list of sed commands separated by NEWLINE characters, as follows:

```
{ command
  command
 }
```

The { can be preceded with blank characters and can be followed with white space. The commands can be preceded by white space. The terminating } must be preceded by a NEWLINE character and can be preceded or followed by <blank>s. The braces may be preceded or followed by <blank>s. The command may be preceded by <blank>s, but may not be followed by <blank>s.

The following table lists the functions, with the maximum number of permissible addresses.

<table>
<thead>
<tr>
<th>Max Address</th>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a\ text</td>
<td>Append by executing N command or beginning a new cycle. Place text on the output before reading the next input line.</td>
</tr>
<tr>
<td>2</td>
<td>b label</td>
<td>Branch to the : command bearing the label. If label is empty, branch to the end of the script. Labels are recognized unique up to eight characters.</td>
</tr>
<tr>
<td>2</td>
<td>c\ text</td>
<td>Change. Delete the pattern space. Place text on the output. Start the next cycle.</td>
</tr>
<tr>
<td>2</td>
<td>d</td>
<td>Delete the pattern space. Start the next cycle.</td>
</tr>
<tr>
<td>Max Address</td>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>2</td>
<td>d</td>
<td>Delete the initial segment of the pattern space through the first new-line. Start the next cycle. (See the \n command below.)</td>
</tr>
<tr>
<td>2</td>
<td>g</td>
<td>Replace the contents of the pattern space by the contents of the hold space.</td>
</tr>
<tr>
<td>2</td>
<td>G</td>
<td>Append the contents of the hold space to the pattern space.</td>
</tr>
<tr>
<td>2</td>
<td>h</td>
<td>Replace the contents of the hold space by the contents of the pattern space.</td>
</tr>
<tr>
<td>2</td>
<td>H</td>
<td>Append the contents of the pattern space to the hold space.</td>
</tr>
<tr>
<td>1</td>
<td>i \text</td>
<td>Insert. Place text on the standard output.</td>
</tr>
<tr>
<td>2</td>
<td>l</td>
<td>/usr/bin/sed: List the pattern space on the standard output in an unambiguous form. Non-printable characters are displayed in octal notation and long lines are folded.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>/usr/xpg4/bin/sed: List the pattern space on the standard output in an unambiguous form. Non-printable characters are displayed in octal notation and long lines are folded. The characters (, \a, \b, \f, \r, \t, and \v) are written as the corresponding escape sequences. Non-printable characters not in that table will be written as one three-digit octal number (with a preceding backslash character) for each byte in the character (most significant byte first). If the size of a byte on the system is greater than nine bits, the format used for non-printable characters is implementation dependent. Long lines are folded, with the point of folding indicated by writing a backslash followed by a NEWLINE; the length at which folding occurs is unspecified, but should be appropriate for the output device. The end of each line is marked with a $.$</td>
</tr>
<tr>
<td>2</td>
<td>n</td>
<td>Copy the pattern space to the standard output if default output is not suppressed. Replace the pattern space with the next line of input.</td>
</tr>
<tr>
<td>Address</td>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>2</td>
<td><code>N</code></td>
<td>Append the next line of input to the pattern space with an embedded new-line. (The current line number changes.) If no next line of input is available, the <code>N</code> command verb shall branch to the end of the script and quit without starting a new cycle and without writing the pattern space.</td>
</tr>
<tr>
<td>2</td>
<td><code>p</code></td>
<td>Print. Copy the pattern space to the standard output.</td>
</tr>
<tr>
<td>2</td>
<td><code>p</code></td>
<td>Copy the initial segment of the pattern space through the first new-line to the standard output.</td>
</tr>
<tr>
<td>1</td>
<td><code>q</code></td>
<td>Quit. Branch to the end of the script. Do not start a new cycle.</td>
</tr>
<tr>
<td>2</td>
<td><code>r file</code></td>
<td>Read the contents of <code>file</code>. Place them on the output before reading the next input line. If <code>file</code> does not exist or cannot be read, it is treated as if it were an empty file, causing no error condition.</td>
</tr>
<tr>
<td>2</td>
<td><code>t label</code></td>
<td>Test. Branch to the : command bearing the <code>label</code> if any substitutions have been made since the most recent reading of an input line or execution of a <code>t</code>. If <code>label</code> is empty, branch to the end of the script.</td>
</tr>
<tr>
<td>2</td>
<td><code>w file</code></td>
<td>Write. Append the pattern space to <code>file</code>. The first occurrence of <code>w</code> will cause <code>file</code> to be cleared. Subsequent invocations of <code>w</code> will append. Each time the <code>sed</code> command is used, <code>file</code> is overwritten.</td>
</tr>
<tr>
<td>2</td>
<td><code>x</code></td>
<td>Exchange the contents of the pattern and hold spaces.</td>
</tr>
<tr>
<td>2</td>
<td><code>! command</code></td>
<td>Don’t. Apply the <code>command</code> (or group, if <code>command</code> is { }) only to lines not selected by the address(es).</td>
</tr>
<tr>
<td>0</td>
<td><code>: label</code></td>
<td>This command does nothing; it bears a <code>label</code> for <code>b</code> and <code>t</code> commands to branch to.</td>
</tr>
<tr>
<td>1</td>
<td><code>=</code></td>
<td>Place the current line number on the standard output as a line.</td>
</tr>
<tr>
<td>2</td>
<td><code>{command-list}</code></td>
<td>Execute <code>command-list</code> only when the pattern space is selected.</td>
</tr>
<tr>
<td>0</td>
<td><code>#</code></td>
<td>An empty command is ignored.</td>
</tr>
<tr>
<td>0</td>
<td><code>#</code></td>
<td>If a <code>#</code> appears as the first character on a line of a script file, then that entire line is treated as a comment, with one exception: if a <code>#</code> appears on the first line and the character after the <code>#</code> is an <code>n</code>, then the default output will be suppressed. The rest of the line after <code>#n</code> is also ignored. A script file must contain at least one non-comment line.</td>
</tr>
</tbody>
</table>
Max Addr | Command (Using *string*) and Description
---|---
2 | `/regular expression/replacement/flags`

Substitute the *replacement* string for instances of the *regular expression* in the pattern space. Any character other than backslash or newline can be used instead of a slash to delimit the RE and the replacement. Within the RE and the replacement, the RE delimiter itself can be used as a literal character if it is preceded by a backslash.

An ampersand (&) appearing in the *replacement* will be replaced by the string matching the RE. The special meaning of & in this context can be suppressed by preceding it by backslash. The characters \n, where n is a digit, will be replaced by the text matched by the corresponding backreference expression. For each backslash (\) encountered in scanning *replacement* from beginning to end, the following character loses its special meaning (if any). It is unspecified what special meaning is given to any character other than &, \ or digits.

A line can be split by substituting a NEWLINE character into it. The application must escape the NEWLINE character in the *replacement* by preceding it with backslash. A substitution is considered to have been performed even if the replacement string is identical to the string that it replaces.

*flags* is zero or more of:

- **n** n = 1 - 512. Substitute for just the n\textsuperscript{th} occurrence of the *regular expression*.
- **g** Global. Substitute for all nonoverlapping instances of the *regular expression* rather than just the first one. If both g and n are specified, the results are unspecified.
- **p** Print the pattern space if a replacement was made.
- **P** Copy the initial segment of the pattern space through the first new-line to the standard output.
- **w** *wfile* Write. Append the pattern space to *wfile* if a replacement was made. The first occurrence of *w* will cause *wfile* to be cleared. Subsequent invocations of *w* will append. Each time the *sed* command is used, *wfile* is overwritten.

2 | `y/ string1 / string2 /`

Transform. Replace all occurrences of characters in *string1* with the corresponding characters in *string2*. *string1* and *string2* must have the same number of characters, or if any of the characters in *string1* appear more than once, the results are undefined. Any character other than backslash or NEWLINE can be used instead of slash to delimit the strings. Within *string1* and *string2*, the delimiter itself can be used as a literal character if it is preceded by a backslash. For example, `y/abc/ABC/` replaces a with A, b with B, and c with C.

See *largefile*(5) for the description of the behavior of *sed* when encountering files greater than or equal to 2 Gbyte (2\textsuperscript{31} bytes).
EXAMPLE 1 An example sed script

This sed script simulates the BSD cat -s command, squeezing excess blank lines from standard input.

```
    sed -n ' # Write non-empty lines.
    /./    {    p
                d
    } # Write a single empty line, then look for more empty lines.
    /^$/   p
    # Get next line, discard the held <newline> (empty line),
    # and look for more empty lines.
    :Empty
    /^$/   {    N
                s/.//
                b Empty
    }
    # Write the non-empty line before going back to search
    # for the first in a set of empty lines.
    ,    p
```

See environ(5) for descriptions of the following environment variables that affect the execution of sed: LC_COLLATE, LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS

The following exit values are returned:
0 Successful completion.
>0 An error occurred.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>/usr/bin/sed</th>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td></td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td></td>
<td>Not enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>/usr/xpg4/bin/sed</th>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td></td>
<td>SUNWxcu4</td>
</tr>
<tr>
<td>CSI</td>
<td></td>
<td>Enabled</td>
</tr>
</tbody>
</table>

SEE ALSO

awk(1), ed(1), grep(1), attributes(5), environ(5), largefile(5), regexp(5), XPG4(5)
**NAME**

sed – stream editor

**SYNOPSIS**

```
sed [-n] [-e script] [-f filename] [filename] ...
```

**DESCRIPTION**

The `sed` utility copies the `filename` (standard input default) to the standard output, edited according to a script of commands.

**OPTIONS**

The following options are supported:

- `-n` Suppresses the default output.
- `-e script` *script* is an edit command for `sed`. If there is just one `-e` option and no `-f` options, the `-e` flag may be omitted.
- `-f filename` Takes the script from `filename`.

**sed Scripts**

`sed` scripts consist of editing commands, one per line, of the following form:

```
[ address [, address ] ] function [ arguments ]
```

In normal operation, `sed` cyclically copies a line of input into a *pattern space* (unless there is something left after a `D` command), sequentially applies all commands with *addresses* matching that pattern space until reaching the end of the script, copies the pattern space to the standard output (except under `-n`), and finally, deletes the pattern space.

Some commands use a *hold space* to save all or part of the pattern space for subsequent retrieval.

An *address* is either:

- a decimal number linecount, which is cumulative across input files;
- a $, which addresses the last input line;
- or a context address, which is a `/regular expression/` as described on the `regexp(5)` manual page, with the following exceptions:

  - `\?RE` In a context address, the construction `\?regular expression?`, where `?` is any character, is identical to `/regular expression/`.

    Note: in the context address `\xabc\xdefx`, the second `x` stands for itself, so that the regular expression is `abcxdef`.

  - `\n` Matches a NEWLINE embedded in the pattern space.

  - `.` Matches any character except the NEWLINE ending the pattern space.

  - `null` A command line with no address selects every pattern space.

  - `address` Selects each pattern space that matches.

  - `address1 , address2` Selects the inclusive range from the first pattern space matching `address1` to the first pattern space matching `address2`.
Selects only one line if \textit{address1} is greater than or equal to \textit{address2}.

\textbf{Comments}

If the first nonwhite character in a line is a `#` (pound sign), \texttt{sed} treats that line as a comment, and ignores it. If, however, the first such line is of the form:

\begin{verbatim}
#n
\end{verbatim}

\texttt{sed} runs as if the \texttt{-n} flag were specified.

\textbf{Functions}

The maximum number of permissible addresses for each function is indicated in parentheses in the list below.

An argument denoted \texttt{text} consists of one or more lines, all but the last of which end with `\' to hide the NEWLINE. Backslashes in text are treated like backslashes in the replacement string of an \texttt{s} command, and may be used to protect initial SPACE and TAB characters against the stripping that is done on every script line.

An argument denoted \texttt{rfilename} or \texttt{wfilename} must terminate the command line and must be preceded by exactly one SPACE. Each \texttt{wfilename} is created before processing begins. There can be at most 10 distinct \texttt{wfilename} arguments.

\begin{itemize}
  \item \texttt{(1) a}\textbackslash
  \texttt{text} Append: place \texttt{text} on the output before reading the next input line.
  \item \texttt{(2) b label} Branch to the `:+' command bearing the \textit{label}. Branch to the end of the script if \textit{label} is empty.
  \item \texttt{(2) c}\textbackslash
  \texttt{text} Change: delete the pattern space. With 0 or 1 address or at the end of a 2 address range, place \texttt{text} on the output. Start the next cycle.
  \item \texttt{(2) d} Delete the pattern space. Start the next cycle.
  \item \texttt{(2) D} Delete the initial segment of the pattern space through the first NEWLINE. Start the next cycle.
  \item \texttt{(2) g} Replace the contents of the pattern space by the contents of the hold space.
  \item \texttt{(2) G} Append the contents of the hold space to the pattern space.
  \item \texttt{(2) h} Replace the contents of the hold space by the contents of the pattern space.
  \item \texttt{(2) H} Append the contents of the pattern space to the hold space.
  \item \texttt{(1) i}\textbackslash
  \texttt{text} Insert: place \texttt{text} on the standard output.
\end{itemize}
List the pattern space on the standard output in an unambiguous form. Non-printing characters are spelled in two digit ASCII and long lines are folded.

Copy the pattern space to the standard output. Replace the pattern space with the next line of input.

Append the next line of input to the pattern space with an embedded newline. (The current line number changes.)

Print: copy the pattern space to the standard output.

Copy the initial segment of the pattern space through the first NEWLINE to the standard output.

Quit: branch to the end of the script. Do not start a new cycle.

Read the contents of $filename$. Place them on the output before reading the next input line.

Substitute the replacement string for instances of the regular expression in the pattern space. Any character may be used instead of `/`. For a fuller description see regexp(5). flags is zero or more of:

- $n$ $n=1-512$. Substitute for just the $n$th occurrence of the regular expression.
- $g$ Global: substitute for all nonoverlapping instances of the regular expression rather than just the first one.
- $p$ Print the pattern space if a replacement was made.
- $w wfilename$ Write: append the pattern space to $wfilename$ if a replacement was made.

Test: branch to the `:` command bearing the label if any substitutions have been made since the most recent reading of an input line or execution of a `t`. If label is empty, branch to the end of the script.

Write: append the pattern space to $wfilename$.

Exchange the contents of the pattern and hold spaces.

Transform: replace all occurrences of characters in $string1$ with the corresponding character in $string2$. The lengths of $string1$ and $string2$ must be equal.

Do not: apply the function (or group, if function is `(`) only to lines not selected by the address(es).

This command does nothing; it bears a label for $b$ and $t$ commands to branch to. Note: the maximum length of label is seven characters.
| (1) = | Place the current line number on the standard output as a line. |
| (2) { | Execute the following commands through a matching ‘}’ only when the pattern space is selected. Commands are separated by ‘;’. |
| (0) | An empty command is ignored. |

Large Files
See largefile(5) for the description of the behavior of sed when encountering files greater than or equal to 2 Gbyte (2^31 bytes).

DIAGNOSTICS
Too many commands
The command list contained more than 200 commands.

Too much command text
The command list was too big for sed to handle. Text in the a, c, and i commands, text read in by r commands, addresses, regular expressions and replacement strings in s commands, and translation tables in y commands all require sed to store data internally.

Command line too long
A command line was longer than 4000 characters.

Too many line numbers
More than 256 decimal number linecounts were specified as addresses in the command list.

Too many files in w commands
More than 10 different files were specified in w commands or w options for s commands in the command list.

Too many labels
More than 50 labels were specified in the command list.

Unrecognized command
A command was not one of the ones recognized by sed.

Extra text at end of command
A command had extra text after the end.

Illegal line number
An address was neither a decimal number linecount, a $, nor a context address.

Space missing before filename
There was no space between an r or w command, or the w option for a s command, and the filename specified for that command.

Too many ‘s
There were more { than } in the list of commands to be executed.

Too many }’s
There were more } than { in the list of commands to be executed.

No addresses allowed
A command that takes no addresses had an address specified.
Only one address allowed
  A command that takes one address had two addresses specified.
"\digit" out of range
  The number in a \n item in a regular expression or a replacement string in an s
  command was greater than 9.
Bad number
  One of the endpoints in a range item in a regular expression (that is, an item of the
  form \{n\} or \{n,m\}) was not a number.
Range endpoint too large
  One of the endpoints in a range item in a regular expression was greater than 255.
More than 2 numbers given in \{ \}
  More than two endpoints were given in a range expression.
} expected after \n
  A \n appeared in a range expression and was not followed by a }.
First number exceeds second in \{ \}
  The first endpoint in a range expression was greater than the second.
Illegal or missing delimiter
  The delimiter at the end of a regular expression was absent.
( \) imbalance
  There were more \( than \), or more \) than \(, in a regular expression.
[ ] imbalance
  There were more \{ than \}, or more \} than \[, in a regular expression.
First RE may not be null
  The first regular expression in an address or in an s command was null (empty).
Ending delimiter missing on substitution
  The ending delimiter in an s command was absent.
Ending delimiter missing on string
  The ending delimiter in a y command was absent.
Transform strings not the same size
  The two strings in a y command were not the same size.
Suffix too large - 512 max
  The suffix in an s command, specifying which occurrence of the regular expression
  should be replaced, was greater than 512.
Label too long
  A label in a command was longer than 8 characters.
Duplicate labels
  The same label was specified by more than one : command.
File name too long
   The filename specified in a r or w command, or in the w option for a s command,
   was longer than 1024 characters.

Output line too long
   An output line was longer than 4000 characters long.

Too many appends or reads after line n
   More than 20 a or r commands were to be executed for line n.

Hold space overflowed.
   More than 4000 characters were to be stored in the hold space.

FILES
   usr/ucb/sed
   BSD sed

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

SEE ALSO
   awk(1), grep(1), lex(1), attributes(5), largefile(5), regexp(5)

BUGS
   There is a combined limit of 200 -e and -f arguments. In addition, there are various
   internal size limits which, in rare cases, may overflow. To overcome these limitations,
   either combine or break out scripts, or use a pipeline of sed commands.
NAME  set, unset, setenv, unsetenv, export – shell built-in functions to determine the characteristics for environmental variables of the current shell and its descendents

**sh**

```sh
set [-aefhkntuvx] [argument]... 
unset [name...] 
export [name...] 

**csh**

```csh
set [var [= value]] 
set var [n] = word 
unset pattern 
setenv [VAR [word]] 
unsetenv variable 
```

**ksh**

```ksh
set [aefhkmnopstuvx] [o option]... [A name] [arg...] 
unset [-f] name... 
**export** [name [=value]]... 
```

**sh**

The set built-in command has the following options:

- - Do not change any of the flags; useful in setting $1 to $.
- a Mark variables which are modified or created for export.
- e Exit immediately if a command exits with a non-zero exit status.
- f Disable file name generation.
- h Locate and remember function commands as functions are defined (function commands are normally located when the function is executed).
- k All keyword arguments are placed in the environment for a command, not just those that precede the command name.
- n Read commands but do not execute them.
- t Exit after reading and executing one command.
- u Treat unset variables as an error when substituting.
- v Print shell input lines as they are read.
- x Print commands and their arguments as they are executed.

Using + rather than - causes these flags to be turned off. These flags can also be used upon invocation of the shell. The current set of flags may be found in $=. The remaining arguments are positional parameters and are assigned, in order, to $1, $2, . . . . If no arguments are given the values of all names are printed.

For each name, unset removes the corresponding variable or function value. The variables PATH, PS1, PS2, MAILCHECK, and IF cannot be unset.
With the `export` built-in, the given *names* are marked for automatic export to the *environment* of subsequently executed commands. If no arguments are given, variable names that have been marked for export during the current shell’s execution are listed. Function names are *not* exported.

With no arguments, `set` displays the values of all shell variables. Multiword values are displayed as a parenthesized list. With the `var` argument alone, `set` assigns an empty (null) value to the variable `var`. With arguments of the form `var = value` `set` assigns `value` to `var`, where `value` is one of:

- *word* A single word (or quoted string).
- *(wordlist)* A space-separated list of words enclosed in parentheses.

Values are command and filename expanded before being assigned. The form `set var[n]=word` replaces the `n`'th word in a multiword value with `word`.

`unset` removes variables whose names match (filename substitution) `pattern`. All variables are removed by `‘unset *’`; this has noticeably distasteful side effects.

With no arguments, `setenv` displays all environment variables. With the `VAR` argument, `setenv` sets the environment variable `VAR` to an empty (null) value. (By convention, environment variables are normally given upper-case names.) With both `VAR` and `word` arguments specified, `setenv` sets `VAR` to `word`, which must be either a single word or a quoted string. The `PATH` variable can take multiple `word` arguments, separated by colons (see EXAMPLES). The most commonly used environment variables, `USER`, `TERM`, and `PATH`, are automatically imported to and exported from the `csh` variables `user`, `term`, and `path`. Use `setenv` if you need to change these variables. In addition, the shell sets the `PWD` environment variable from the `csh` variable `cwd` whenever the latter changes.

The environment variables `LC_CTYPE`, `LC_MESSAGES`, `LC_TIME`, `LC_COLLATE`, `LC_NUMERIC`, and `LC_MONETARY` take immediate effect when changed within the `C` shell. See `environ(5)` for descriptions of these environment variables.

`unsetenv` removes `variable` from the environment. As with `unset`, pattern matching is not performed.

The flags for the `set` built-in have meaning as follows:

- `-A` Array assignment. Unset the variable `name` and assign values sequentially from the list `arg`. If `+A` is used, the variable `name` is not unset first.
- `-a` All subsequent variables that are defined are automatically exported.
- `-e` If a command has a non-zero exit status, execute the `ERR` trap, if set, and `exit`. This mode is disabled while reading profiles.
- `-f` Disables file name generation.
- `-h` Each command becomes a tracked alias when first encountered.
-k All variable assignment arguments are placed in the environment for a command, not just those that precede the command name.

-m Background jobs will run in a separate process group and a line will print upon completion. The exit status of background jobs is reported in a completion message. On systems with job control, this flag is turned on automatically for interactive shells.

-n Read commands and check them for syntax errors, but do not execute them. Ignored for interactive shells.

-o The following argument can be one of the following option names:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>allexport</td>
<td>Same as -a.</td>
</tr>
<tr>
<td>errexit</td>
<td>Same as -e.</td>
</tr>
<tr>
<td>bgnice</td>
<td>All background jobs are run at a lower priority. This is the default mode. emacs Puts you in an emacs style in-line editor for command entry.</td>
</tr>
<tr>
<td>gmacs</td>
<td>Puts you in a gmacs style in-line editor for command entry.</td>
</tr>
<tr>
<td>ignoreeof</td>
<td>The shell will not exit on end-of-file. The command exit must be used.</td>
</tr>
<tr>
<td>keyword</td>
<td>Same as -k.</td>
</tr>
<tr>
<td>markdirs</td>
<td>All directory names resulting from file name generation have a trailing / appended.</td>
</tr>
<tr>
<td>monitor</td>
<td>Same as -m.</td>
</tr>
<tr>
<td>noclobber</td>
<td>Prevents redirection &gt; from truncating existing files. Require &gt;</td>
</tr>
<tr>
<td>noexec</td>
<td>Same as -n.</td>
</tr>
<tr>
<td>noglob</td>
<td>Same as -f.</td>
</tr>
<tr>
<td>nolog</td>
<td>Do not save function definitions in history file.</td>
</tr>
<tr>
<td>nounset</td>
<td>Same as -u.</td>
</tr>
<tr>
<td>privileged</td>
<td>Same as -p.</td>
</tr>
<tr>
<td>verbose</td>
<td>Same as -v.</td>
</tr>
<tr>
<td>trackall</td>
<td>Same as -h.</td>
</tr>
<tr>
<td>vi</td>
<td>Puts you in insert mode of a vi style in-line editor until you hit escape character 033. This puts you in control mode. A return sends the line.</td>
</tr>
<tr>
<td>viraw</td>
<td>Each character is processed as it is typed in vi mode.</td>
</tr>
</tbody>
</table>
xtrace       Same as -x.

If no option name is supplied then the current option settings are printed.

- p         Disables processing of the $HOME/.profile file and uses the file
            /etc/suid_profile instead of the ENV file. This mode is on whenever
            the effective uid is not equal to the real uid, or when the effective gid is not
            equal to the real gid. Turning this off causes the effective uid and gid to be
            set to the real uid and gid.

- s         Sort the positional parameters lexicographically.

- t         Exit after reading and executing one command.

- u         Treat unset parameters as an error when substituting.

- v         Print shell input lines as they are read.

- x         Print commands and their arguments as they are executed.

−          Turns off -x and -v flags and stops examining arguments for flags.

−          Do not change any of the flags; useful in setting $1 to a value beginning
            with −. If no arguments follow this flag then the positional parameters are
            unset.

Using + rather than − causes these flags to be turned off. These flags can also be used
upon invocation of the shell. The current set of flags may be found in $−. Unless -A is
specified, the remaining arguments are positional parameters and are assigned, in
order, to $1 $2 . . . . If no arguments are given then the names and values of all
variables are printed on the standard output.

The variables given by the list of names are unassigned, i.e., their values and attributes
are erased. readonly variables cannot be unset. If the -f, flag is set, then the names
refer to function names. Unsetting ERRNO, LINENO, MAILCHECK, OPTARG, OPTIND,
RANDOM, SECONDS, TMOUT, and _ removes their special meaning even if they are
subsequently assigned.

When using unset, the variables given by the list of names are unassigned, i.e., their
values and attributes are erased. readonly variables cannot be unset. If the -f, flag is
set, then the names refer to function names. Unsetting ERRNO, LINENO,
MAILCHECK, OPTARG, OPTIND, RANDOM, SECONDS, TMOUT, and _ removes their
special meaning even if they are subsequently assigned.

With the export built-in, the given names are marked for automatic export to the
environment of subsequently-executed commands.

On this man page, ksh(1) commands that are preceded by one or two * (asterisks) are
treated specially in the following ways:

1. Variable assignment lists preceding the command remain in effect when the
   command completes.
1. I/O redirections are processed after variable assignments.
2. Errors cause a script that contains them to abort.
3. Words, following a command preceded by ** that are in the format of a variable assignment, are expanded with the same rules as a variable assignment. This means that tilde substitution is performed after the = sign and word splitting and file name generation are not performed.

The following example sets the PATH variable to search for files in the /bin, /usr/bin, /usr/sbin, and /usr/ucb/bin directories, in that order.

```
setenv PATH "/bin:/usr/bin:/usr/sbin:usr/ucb/bin"
```

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

**SEE ALSO**
csh(1), ksh(1), read(1), sh(1), typeset(1), attributes(5), environ(5)
set, unset – set and unset local or global environment variables

**SYNOPSIS**

```bash
set [-l variable [=value]] ...
set [-e variable [=value]] ...
set [-f file variable [=value]] ...
unset -l variable...
unset -f file variable...
```

**DESCRIPTION**

The `set` command sets `variable` in the environment, or adds `variable=value` to `file`. If `variable` is not equated to a value, `set` expects the value to be on `stdin`. The `unset` command removes `variable`. Note that the FMLI predefined, read-only variables (such as `ARG1`), may not be set or unset.

Note that at least one of the above options must be used for each variable being set or unset. If you set a variable with the `-f filename` option, you must thereafter include `filename` in references to that variable. For example, `$({(file) VARIABLE})`.

FMLI inherits the UNIX environment when invoked.

**OPTIONS**

- `-l` Sets or unsets the specified variable in the local environment. Variables set with `-l` will not be inherited by processes invoked from FMLI.

- `-e` Sets the specified variable in the UNIX environment. Variables set with `-e` will be inherited by any processes started from FMLI. Note that these variables cannot be `unset`.

- `-f file` Sets or unsets the specified variable in the global environment. The argument `file` is the name, or pathname, of a file containing lines of the form `variable=value`. `file` will be created if it does not already exist. Note that no space intervenes between `-f` and `file`.

**EXAMPLES**

**EXAMPLE 1** A sample output of `set` command.

Storing a selection made in a menu:

```bash
name=Selection 2
action='set -l SELECTION=2' 'close
```

**NOTES**

Variables set to be available to the UNIX environment (those set using the `-e` option) can only be set for the current `fml` process and the processes it calls.

When using the `-f` option, unless `file` is unique to the process, other users of FMLI on the same machine will be able to expand these variables, depending on the read/write permissions on `file`.

A variable set in one frame may be referenced or unset in any other frame. This includes local variables.

**ATTRIBUTES**

See `attributes(5)` for descriptions of the following attributes:
<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

**SEE ALSO**  
env(1), sh(1), attributes(5)
setcolor(1F)

NAME setcolor – redefine or create a color

SYNOPSIS setcolor color red_level green_level blue_level

DESCRIPTION The setcolor command takes four arguments: color, which must be a string naming the color; and the arguments red_level, green_level, and blue_level, which must be integer values defining, respectively, the intensity of the red, green, and blue components of color. Intensities must be in the range of 0 to 1000. If you are redefining an existing color, you must use its current name (default color names are: black, blue, green, cyan, red, magenta, yellow, and white). setcolor returns the color’s name string.

EXAMPLES EXAMPLE 1 A sample output of setcolor command.
The following is an example of the arguments that setcolor takes:

'setcolor blue 100 24 300'

BUILTIN FMLI

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO attributes(5)
NAME  setfacl – modify the Access Control List (ACL) for a file or files

SYNOPSIS

setfacl [-r] -s acl_entries file
setfacl [-r] -m a acl_entries file
setfacl [-r] -f acl_file

DESCRIPTION

For each file specified, setfacl will either replace its entire ACL, including the default ACL on a directory, or it will add, modify, or delete one or more ACL entries, including default entries on directories.

When the setfacl command is used, it may result in changes to the file permission bits. When the user ACL entry for the file owner is changed, the file owner class permission bits will be modified. When the group ACL entry for the file group class is changed, the file group class permission bits will be modified. When the other ACL entry is changed, the file other class permission bits will be modified.

If you use the chmod(1) command to change the file group owner permissions on a file with ACL entries, both the file group owner permissions and the ACL mask are changed to the new permissions. Be aware that the new ACL mask permissions may change the effective permissions for additional users and groups who have ACL entries on the file.

A directory may contain default ACL entries. If a file or directory is created in a directory that contains default ACL entries, the newly created file will have permissions generated according to the intersection of the default ACL entries and the permissions requested at creation time. The umask(1) will not be applied if the directory contains default ACL entries. If a default ACL is specified for a specific user (or users), the file will have a regular ACL created. Otherwise, only the mode bits will be initialized according to the intersection described above. The default ACL should be thought of as the maximum discretionary access permissions that may be granted.

For the -m and -s options, acl_entries are one or more comma-separated ACL entries.

 acl_entries Syntax

For the -m and -s options, acl_entries are one or more comma-separated ACL entries.

An ACL entry consists of the following fields separated by colons:

entry_type   Type of ACL entry on which to set file permissions. For example, entry_type can be user (the owner of a file) or mask (the ACL mask).

uid or gid   User name or user identification number. Or, group name or group identification number.

perms   Represents the permissions that are set on entry_type. perms can be indicated by the symbolic characters rwx or a number (the same permissions numbers used with the chmod command).

The following table shows the valid ACL entries (default entries may only be specified for directories):
For the `-d` option, *acl_entries* are one or more comma-separated ACL entries without permissions. Note that the entries for file owner, file group owner, ACL mask, and others may not be deleted.

**OPTIONS**

The options have the following meaning:

- `-d acl_entries` Deletes one or more entries from the file. The entries for the file owner, the file group owner, and others may not be deleted from the ACL. Notice that deleting an entry does not necessarily have the same effect as removing all permissions from the entry.

- `-f acl_file` Set a file’s ACL with the ACL entries contained in the file named *acl_file*. The same constraints on specified entries hold as with the `-a` option. The entries are not required to be in any specific order in the file. Also, if you specify a dash ‘-’ for *acl_file*, standard input is used to set the file’s ACL.
The character "#" in acl_file may be used to indicate a comment. All characters, starting with the "#" until the end of the line, will be ignored. Note that if the acl_file has been created as the output of the getfacl(1) command, any effective permissions, which will follow a "#", will be ignored.

-m acl_entries

Adds one or more new ACL entries to the file, and/or modifies one or more existing ACL entries on the file. If an entry already exists for a specified uid or gid, the specified permissions will replace the current permissions. If an entry does not exist for the specified uid or gid, an entry will be created. When using the -m option to modify a default ACL, you must specify a complete default ACL (user, group, other, mask, and any additional entries) the first time.

-r

Recalculates the permissions for the ACL mask entry. The permissions specified in the ACL mask entry are ignored and replaced by the maximum permissions necessary to grant the access to all additional user, file group owner, and additional group entries in the ACL. The permissions in the additional user, file group owner, and additional group entries are left unchanged.

-s acl_entries

Sets a file's ACL. All old ACL entries are removed and replaced with the newly specified ACL. The entries need not be in any specific order. They will be sorted by the command before being applied to the file.

Required entries:
- Exactly one user entry specified for the file owner.
- Exactly one group entry for the file group owner.
- Exactly one other entry specified.

If there are additional user and group entries:
- Exactly one mask entry specified for the ACL mask that indicates the maximum permissions allowed for users (other than the owner) and groups.
- Must not be duplicate user entries with the same uid.
- Must not be duplicate group entries with the same gid.

If file is a directory, the following default ACL entries may be specified:
- Exactly one default user entry for the file owner.
- Exactly one default group entry for the file group owner.
- Exactly one default mask entry for the ACL mask.
- Exactly one default other entry.
There may be additional default user entries and additional default group entries specified, but there may not be duplicate additional default user entries with the same uid, or duplicate default group entries with the same gid.

**EXAMPLE 1** Adding read permission only

The following example adds one ACL entry to file abc, which gives user shea read permission only.

```
setfacl -m user:shea:r -- abc
```

**EXAMPLE 2** Replacing a file's entire ACL

The following example replaces the entire ACL for the file abc, which gives shea read access, the file owner all access, the file group owner read access only, the ACL mask read/write access, and others no access.

```
setfacl -s user:shea:rwx,user::rwx,group::rw,mask:r--,other:---- abc
```

Notice that after this command, the file permission bits are rwxr------. Even though the file group owner was set with read/write permissions, the ACL mask entry limits it to have only read permissions. The mask entry also specifies the maximum permissions available to all additional user and group ACL entries. Once again, even though the user shea was set with all access, the mask limits it to have only read permissions. The ACL mask entry is a quick way to limit or open access to all the user and group entries in an ACL. For example, by changing the mask entry to read/write, both the file group owner and user shea would be given read/write access.

**EXAMPLE 3** Setting the same ACL on two files

The following example sets the same ACL on file abc as the file xyz.

```
getfacl xyz | setfacl -f -- abc
```

**FILES**

/etc/passwd  password file

/etc/group  group file

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

**SEE ALSO**

setfacl(1), umask(1), aclcheck(3SEC), aclsort(3SEC), group(4), passwd(4), attributes(5), chmod(1)
NAME
setpgrep – set process group ID

SYNOPSIS
setpgrep command [arg...]

DESCRIPTION
If the current process is not already a session leader, the setpgrep utility sets the
process group ID and session ID to the current process ID and does an exec() of
command and its argument(s), if any.

OPERANDS
The following operands are supported:
command The name of a command to be invoked.
arg An option or argument to command.

EXIT STATUS
The following exit values are returned:
1 Error executing the setpgrep utility or during exec() of
command.

Otherwise, the exit status will be that of command.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO
exec(2), setpgrep(2), attributes(5)
**NAME**
sftp – secure file transfer program

**SYNOPSIS**
```
sftp [-vC] [-o ssh_option] [hostname | user@hostname]
```

**DESCRIPTION**
The *sftp* utility is an interactive file transfer program with a user interface similar to *ftp*(1) that uses the *ssh*(1) command to create a secure connection to the server.

*sftp* implements the SSH File Transfer Protocol as defined in IETF draft-ietf-secsh-filexfer. There is no relationship between the protocol used by *sftp* and the FTP protocol (RFC959) provided by *ftp*(1).

**OPTIONS**
The following options are supported:
- `-v`
  Raises logging level. This option is also passed to *ssh*(1).
- `-C`
  Enables compression, using the `-C` flag in *ssh*(1).
- `-o ssh_option`
  Specifies an option to be directly passed to *ssh*(1).

**OPERANDS**
The following operands are supported:
- `hostname | user@hostname`
  The name of the host to which *sftp* connects and logs into.

**INTERACTIVE COMMANDS**
Once in interactive mode, *sftp* understands a set of commands similar to those of *ftp*(1). Commands are case insensitive and pathnames may be enclosed in quotes if they contain spaces.

- `cd path`
  Changes remote directory to *path*.

- `lcd path`
  Changes local directory to *path*.

- `chgrp grp path`
  Changes group of file *path* to *grp*. *grp* must be a numeric GID.

- `chmod mode path`
  Changes permissions of file *path* to *mode*.

- `chown own path`
  Changes owner of file *path* to *own*. *own* must be a numeric UID.

- `help`
  Displays help text.

- `get [flags] remote-path [local-path]`
  Retrieves the remote-path and stores it on the local machine. If the local path name is not specified, it is given the same name it has on the remote machine. If the `-P` flag is specified, then the file’s full permission and access time are copied too.

- `lls [ls-options [path]]`
  Displays local directory listing of either *path* or current directory if *path* is not specified.
Sftp

1mkdir path
   Creates local directory specified by path.

lpwd
   Prints local working directory.

ls [path]
   Displays remote directory listing of either path or current directory if path is not specified.

lumask umask
   Sets local umask to umask.

mkdir path
   Creates remote directory specified by path.

put [flags] local-path [local-path]
   Uploads local-path and stores it on the remote machine. If the remote path name is not specified, it is given the same name it has on the local machine. If the -P flag is specified, then the file’s full permission and access time are copied too.

pwd
   Displays remote working directory.

exit
   Quits sftp.

quit
   Quits sftp.

rename oldpath newpath
   Renames remote file from oldpath to newpath.

rmdir path
   Removes remote directory specified by path.

rm path
   Deletes remote file specified by path.

! command
   Executes command in local shell.

! 
   Escapes to local shell.

? 
   Synonym for help.

EXIT STATUS
The following exit values are returned:

   0   Successful completion.
   1   An error occurred.

ATTRIBUTES
See attributes(5) for descriptions of the following attributes:
To view license terms, attribution, and copyright for OpenSSH, the default path is /var/sadm/pkg/SUNWsshdr/install/copyright. If the Solaris operating environment has been installed anywhere other than the default, modify the given path to access the file at the installed location.

AUTHOR Damien Miller
NAME  sh, jsh – standard and job control shell and command interpreter

SYNOPSIS  
/usr/bin/sh [-acefhiknprstuvx] [argument...]
/usr/xpg4/bin/sh [± abCefhikmnoprstuvx] [± o option...] [-c string] [arg...]
/usr/bin/jsh [-acefhiknprstuvx] [argument...]

DESCRIPTION  The /usr/bin/sh utility is a command programming language that executes commands read from a terminal or a file.

The /usr/xpg4/bin/sh utility is a standards compliant shell. This utility provides all the functionality of ksh(1), except in cases discussed in ksh(1) where differences in behavior exist.

The jsh utility is an interface to the shell that provides all of the functionality of sh and enables job control (see Job Control section below).

Arguments to the shell are listed in the Invocation section below.

Definitions  A blank is a tab or a space. A name is a sequence of ASCII letters, digits, or underscores, beginning with a letter or an underscore. A parameter is a name, a digit, or any of the characters *,$, @, #, ?, −, $, and !.

Commands  A simple-command is a sequence of non-blank words separated by blanks. The first word specifies the name of the command to be executed. Except as specified below, the remaining words are passed as arguments to the invoked command. The command name is passed as argument 0 (see exec(2)). The value of a simple-command is its exit status if it terminates normally, or (octal) 200+status if it terminates abnormally. See signal(3HEAD) for a list of status values.

A pipeline is a sequence of one or more commands separated by |. The standard output of each command but the last is connected by a pipe(2) to the standard input of the next command. Each command is run as a separate process. The shell waits for the last command to terminate. The exit status of a pipeline is the exit status of the last command in the pipeline.

A list is a sequence of one or more pipelines separated by :, & & , or | | , and optionally terminated by ; or & . Of these four symbols, ; and & have equal precedence, which is lower than that of & & and | | . The symbols & & and | | also have equal precedence. A semicolon (;) causes sequential execution of the preceding pipeline, that is, the shell waits for the pipeline to finish before executing any commands following the semicolon. An ampersand (&) causes asynchronous execution of the preceding pipeline, that is, the shell does not wait for that pipeline to finish. The symbol & & ( | | ) causes the list following it to be executed only if the preceding pipeline returns a zero (non-zero) exit status. An arbitrary number of newlines may appear in a list, instead of semicolons, to delimit commands.
A *command* is either a *simple-command* or one of the following. Unless otherwise stated, the value returned by a command is that of the last *simple-command* executed in the command.

```bash
for name [ in word ... ] do list done
```

Each time a *for* command is executed, *name* is set to the next *word* taken from the *in word* list. If *in word* ... is omitted, then the *for* command executes the *do list* once for each positional parameter that is set (see Parameter Substitution section below). Execution ends when there are no more words in the list.

```bash
case word in [ pattern [ | pattern ] list ; ; ]... esac
```

A *case* command executes the *list* associated with the first *pattern* that matches *word*. The form of the patterns is the same as that used for file-name generation (see File Name Generation section), except that a slash, a leading dot, or a dot immediately following a slash need not be matched explicitly.

```bash
if list ; then list ; [ elif list ; then list ; ]...[ else list ; ] fi
```

The *list* following *if* is executed and, if it returns a zero exit status, the *list* following the first *then* is executed. Otherwise, the *list* following *elif* is executed and, if its value is zero, the *list* following the next *then* is executed. Failing that, the *else list* is executed. If no *else list* or *then list* is executed, then the *if* command returns a zero exit status.

```bash
while list do list done
```

A *while* command repeatedly executes the *while list* and, if the exit status of the last command in the list is zero, executes the *do list*; otherwise the loop terminates. If no commands in the *do list* are executed, then the *while* command returns a zero exit status; *until* may be used in place of *while* to negate the loop termination test.

```bash
(list)
```

Execute *list* in a sub-shell.

```bash
{ list; }
```

*list* is executed in the current (that is, parent) shell. The *{ * must be followed by a space.

```bash
name ( ) { list; }
```

Define a function which is referenced by *name*. The body of the function is the *list* of commands between *{ * and *}*. The *{ * must be followed by a space. Execution of functions is described below (see Execution section). The *{ * and *} * are unnecessary if the body of the function is a *command* as defined above, under Commands.

The following words are only recognized as the first word of a command and when not quoted:
A word beginning with # causes that word and all the following characters up to a newline to be ignored.

The shell reads commands from the string between two grave accents (``) and the standard output from these commands may be used as all or part of a word. Trailing newlines from the standard output are removed.

No interpretation is done on the string before the string is read, except to remove backslashes (\) used to escape other characters. Backslashes may be used to escape a grave accent (`) or another backslash (\) and are removed before the command string is read. Escaping grave accents allows nested command substitution. If the command substitution lies within a pair of double quotes (" ... "), a backslash used to escape a double quote (\") will be removed; otherwise, it will be left intact.

If a backslash is used to escape a newline character (\newline), both the backslash and the newline are removed (see the later section on Quoting). In addition, backslashes used to escape dollar signs ($), "", newline, and $ are left intact when the command string is read.

The character $ is used to introduce substitutable parameters. There are two types of parameters, positional and keyword. If parameter is a digit, it is a positional parameter. Positional parameters may be assigned values by set. Keyword parameters (also known as variables) may be assigned values by writing:

```
name=value [ name=value ] ...
```

Pattern-matching is not performed on value. There cannot be a function and a variable with the same name.

```
${parameter}
```

The value, if any, of the parameter is substituted. The braces are required only when parameter is followed by a letter, digit, or underscore that is not to be interpreted as part of its name. If parameter is * or @, all the positional parameters, starting with $1, are substituted (separated by spaces). Parameter $0 is set from argument zero when the shell is invoked.

```
${parameter:word}
```

If parameter is set and is non-null, substitute its value; otherwise substitute word.

```
${parameter:=word}
```

If parameter is not set or is null set it to word; the value of the parameter is substituted. Positional parameters may not be assigned in this way.
If `parameter` is set and is non-null, substitute its value; otherwise, print `word` and exit from the shell. If `word` is omitted, the message “parameter null or not set” is printed.

If `parameter` is set and is non-null, substitute `word`; otherwise substitute nothing.

In the above, `word` is not evaluated unless it is to be used as the substituted string, so that, in the following example, `pwd` is executed only if `d` is not set or is null:

```bash
echo ${d:−'pwd'}
```

If the colon (:) is omitted from the above expressions, the shell only checks whether `parameter` is set or not.

The following parameters are automatically set by the shell.

- `#` The number of positional parameters in decimal.
- `-` Flags supplied to the shell on invocation or by the `set` command.
- `?` The decimal value returned by the last synchronously executed command.
- `$` The process number of this shell.
- `!` The process number of the last background command invoked.

The following parameters are used by the shell. The parameters in this section are also referred to as environment variables.

- `HOME` The default argument (home directory) for the `cd` command, set to the user’s login directory by `login(1)` from the password file (see `passwd(4)`).
- `PATH` The search path for commands (see Execution section below).
- `CDPATH` The search path for the `cd` command.
- `MAIL` If this parameter is set to the name of a mail file and the `MAILPATH` parameter is not set, the shell informs the user of the arrival of mail in the specified file.
- `MAILCHECK` This parameter specifies how often (in seconds) the shell will check for the arrival of mail in the files specified by the `MAILPATH` or `MAIL` parameters. The default value is 600 seconds (10 minutes). If set to 0, the shell will check before each prompt.
- `MAILPATH` A colon-separated list of file names. If this parameter is set, the shell informs the user of the arrival of mail in any of the specified
files. Each file name can be followed by % and a message that will be printed when the modification time changes. The default message is, you have mail.

PS1
Primary prompt string, by default “ $ ”.

PS2
Secondary prompt string, by default “ > ”.

IFS
Internal field separators, normally space, tab, and newline (see Blank Interpretation section).

SHACCT
If this parameter is set to the name of a file writable by the user, the shell will write an accounting record in the file for each shell procedure executed.

SHELL
When the shell is invoked, it scans the environment (see Environment section below) for this name.

See environ(5) for descriptions of the following environment variables that affect the execution of sh: LC_CTYPE and LC_MESSAGES.

The shell gives default values to PATH, PS1, PS2, MAILCHECK, and IFS. Default values for HOME and MAIL are set by login(1).

After parameter and command substitution, the results of substitution are scanned for internal field separator characters (those found in IFS) and split into distinct arguments where such characters are found. Explicit null arguments ("" or ' ') are retained. Implicit null arguments (those resulting from parameters that have no values) are removed.

A command’s input and output may be redirected using a special notation interpreted by the shell. The following may appear anywhere in a simple-command or may precede or follow a command and are not passed on as arguments to the invoked command.

Note: Parameter and command substitution occurs before word or digit is used.

<word
Use file word as standard input (file descriptor 0).

>word
Use file word as standard output (file descriptor 1). If the file does not exist, it is created; otherwise, it is truncated to zero length.

>>word
Use file word as standard output. If the file exists, output is appended to it by first seeking to the EOF. Otherwise, the file is created.

< >word
Open file word for reading and writing as standard input.

<<[–]word
After parameter and command substitution is done on word, the shell input is read up to the first line that literally matches the resulting word, or to an EOF. If, however, the hyphen (–) is appended to <:
1. leading tabs are stripped from *word* before the shell input is read (but after parameter and command substitution is done on *word*);
2. leading tabs are stripped from the shell input as it is read and before each line is compared with *word*; and
3. shell input is read up to the first line that literally matches the resulting *word*, or to an EOF.

If any character of *word* is quoted (see Quoting section later), no additional processing is done to the shell input. If no characters of *word* are quoted:

1. parameter and command substitution occurs;
2. (escaped) \newlines are removed; and
3. \ must be used to quote the characters \, $, and ‘.

The resulting document becomes the standard input.

<&digit Use the file associated with file descriptor *digit* as standard input. Similarly for the standard output using >&digit.

<&= The standard input is closed. Similarly for the standard output using >&=.

If any of the above is preceded by a digit, the file descriptor which will be associated with the file is that specified by the digit (instead of the default 0 or 1). For example:

... 2>&1

associates file descriptor 2 with the file currently associated with file descriptor 1.

The order in which redirections are specified is significant. The shell evaluates redirections left-to-right. For example:

... 1>&xx 2>&1

first associates file descriptor 1 with file *xxx*. It associates file descriptor 2 with the file associated with file descriptor 1 (that is, *xxx*). If the order of redirections were reversed, file descriptor 2 would be associated with the terminal (assuming file descriptor 1 had been) and file descriptor 1 would be associated with file *xxx*.

Using the terminology introduced on the first page, under Commands, if a *command* is composed of several *simple commands*, redirection will be evaluated for the entire *command* before it is evaluated for each *simple command*. That is, the shell evaluates redirection for the entire *list*, then each *pipeline* within the *list*, then each *command* within each *pipeline*, then each *list* within each *command*.

If a command is followed by & and the default standard input for the command is the empty file, /dev/null. Otherwise, the environment for the execution of a command contains the file descriptors of the invoking shell as modified by input/output specifications.
Before a command is executed, each command word is scanned for the characters *, ?, and [. If one of these characters appears the word is regarded as a pattern. The word is replaced with alphabetically sorted file names that match the pattern. If no file name is found that matches the pattern, the word is left unchanged. The character . at the start of a file name or immediately following a /, as well as the character / itself, must be matched explicitly.

* Matches any string, including the null string.
? Matches any single character.
[...] Matches any one of the enclosed characters. A pair of characters separated by − matches any character lexically between the pair, inclusive. If the first character following the opening [ is a !, any character not enclosed is matched.

Notice that all quoted characters (see below) must be matched explicitly in a filename.

The following characters have a special meaning to the shell and cause termination of a word unless quoted:

; & ( ) | ^ < > newline space tab

A character may be quoted (that is, made to stand for itself) by preceding it with a backslash (\) or inserting it between a pair of quote marks (’’ or “”). During processing, the shell may quote certain characters to prevent them from taking on a special meaning. Backslashes used to quote a single character are removed from the word before the command is executed. The pair \newline is removed from a word before command and parameter substitution.

All characters enclosed between a pair of single quote marks (’’), except a single quote, are quoted by the shell. Backslash has no special meaning inside a pair of single quotes. A single quote may be quoted inside a pair of double quote marks (for example, ”’”), but a single quote can not be quoted inside a pair of single quotes.

Inside a pair of double quote marks (””), parameter and command substitution occurs and the shell quotes the results to avoid blank interpretation and file name generation. If $* is within a pair of double quotes, the positional parameters are substituted and quoted, separated by quoted spaces ("$1  $2 ... "). However, if $@ is within a pair of double quotes, the positional parameters are substituted and quoted, separated by unquoted spaces ("$1  "$2  ... ). \ quotes the characters , , , (comma), and $.

The pair \newline is removed before parameter and command substitution. If a backslash precedes characters other than , , (comma), $, and newline, then the backslash itself is quoted by the shell.

When used interactively, the shell prompts with the value of PS1 before reading a command. If at any time a newline is typed and further input is needed to complete a command, the secondary prompt (that is, the value of PS2) is issued.
The environment (see environ(5)) is a list of name-value pairs that is passed to an executed program in the same way as a normal argument list. The shell interacts with the environment in several ways. On invocation, the shell scans the environment and creates a parameter for each name found, giving it the corresponding value. If the user modifies the value of any of these parameters or creates new parameters, none of these affects the environment unless the export command is used to bind the shell’s parameter to the environment (see also set -a). A parameter may be removed from the environment with the unset command. The environment seen by any executed command is thus composed of any unmodified name-value pairs originally inherited by the shell, minus any pairs removed by unset, plus any modifications or additions, all of which must be noted in export commands.

The environment for any simple-command may be augmented by prefixing it with one or more assignments to parameters. Thus:

```
TERM=450  command
```

and

```
(export TERM; TERM=450;    command
```

are equivalent as far as the execution of command is concerned if command is not a Special Command. If command is a Special Command, then

```
TERM=450  command
```

will modify the TERM variable in the current shell.

If the -k flag is set, all keyword arguments are placed in the environment, even if they occur after the command name. The following example first prints a=b c and c:

```
echo a=b c
a=b c
set -k
echo a=b c
```

The INTERRUPT and QUIT signals for an invoked command are ignored if the command is followed by &. Otherwise, signals have the values inherited by the shell from its parent, with the exception of signal 11 (but see also the trap command below).

Each time a command is executed, the command substitution, parameter substitution, blank interpretation, input/output redirection, and filename generation listed above are carried out. If the command name matches the name of a defined function, the function is executed in the shell process (note how this differs from the execution of
shell script files, which require a sub-shell for invocation. If the command name does 
not match the name of a defined function, but matches one of the Special 
Commands listed below, it is executed in the shell process.

The positional parameters \$1, \$2,... are set to the arguments of the function. If the 
command name matches neither a Special Command nor the name of a defined 
function, a new process is created and an attempt is made to execute the command via 
exec(2).

The shell parameter \texttt{PATH} defines the search path for the directory containing 
the command. Alternative directory names are separated by a colon (\texttt{:}). The default path 
is /usr/bin. The current directory is specified by a null path name, which can appear 
immediately after the equal sign, between two colon delimiters anywhere in the path 
list, or at the end of the path list. If the command name contains a / the search path is 
not used. Otherwise, each directory in the path is searched for an executable file. If the 
file has execute permission but is not an a.out file, it is assumed to be a file 
containing shell commands. A sub-shell is spawned to read it. A parenthesized 
command is also executed in a sub-shell.

The location in the search path where a command was found is remembered by the 
shell (to help avoid unnecessary execs later). If the command was found in a relative 
directory, its location must be re-determined whenever the current directory changes. 
The shell forgets all remembered locations whenever the \texttt{PATH} variable is changed or 
the \texttt{hash -r} command is executed (see below).

Input/output redirection is now permitted for these commands. File descriptor 1 is 
the default output location. When Job Control is enabled, additional Special 
Commands are added to the shell’s environment (see Job Control section below).

\texttt{:}

No effect; the command does nothing. A zero exit code is returned.

\texttt{. filename}

Read and execute commands from \texttt{filename} and return. The search path specified by 
\texttt{PATH} is used to find the directory containing \texttt{filename}.

\texttt{bg \{\%jobid \ldots\}}

When Job Control is enabled, the \texttt{bg} command is added to the user’s environment 
to manipulate jobs. Resumes the execution of a stopped job in the background. If 
\texttt{\%jobid} is omitted the current job is assumed. (See Job Control section below for 
more detail.)

\texttt{break \{n\}}

Exit from the enclosing for or while loop, if any. If \texttt{n} is specified, break \texttt{n} levels.

\texttt{cd \{argument\}}

Change the current directory to \texttt{argument}. The shell parameter \texttt{HOME} is the default 
\texttt{argument}. The shell parameter \texttt{CDPATH} defines the search path for the directory 
containing \texttt{argument}. Alternative directory names are separated by a colon (\texttt{:}). The 
default path is <null> (specifying the current directory). \texttt{Note:} The current 
directory is specified by a null path name, which can appear immediately after the
equal sign or between the colon delimiters anywhere else in the path list. If argument begins with a / the search path is not used. Otherwise, each directory in the path is searched for argument.

chdir [dir]
chdir changes the shell’s working directory to directory dir. If no argument is given, change to the home directory of the user. If dir is a relative pathname not found in the current directory, check for it in those directories listed in the CDPATH variable. If dir is the name of a shell variable whose value starts with a /, change to the directory named by that value.

continue [n]
Resume the next iteration of the enclosing for or while loop. If n is specified, resume at the n-th enclosing loop.

echo [arguments ...]
The words in arguments are written to the shell’s standard output, separated by space characters. See echo(1) for fuller usage and description.

eval [argument ...]
The arguments are read as input to the shell and the resulting command(s) executed.

exec [argument ...]
The command specified by the arguments is executed in place of this shell without creating a new process. Input/output arguments may appear and, if no other arguments are given, cause the shell input/output to be modified.

exit [n]
Causes the calling shell or shell script to exit with the exit status specified by n. If n is omitted the exit status is that of the last command executed (an EOF will also cause the shell to exit.)

export [name ...]
The given names are marked for automatic export to the environment of subsequently executed commands. If no arguments are given, variable names that have been marked for export during the current shell’s execution are listed. (Variable names exported from a parent shell are listed only if they have been exported again during the current shell’s execution.) Function names are not exported.

fg [%jobid ...]
When Job Control is enabled, the fg command is added to the user’s environment to manipulate jobs. This command resumes the execution of a stopped job in the foreground and also moves an executing background job into the foreground. If %jobid is omitted, the current job is assumed. (See Job Control section below for more detail.)

getopts
Use in shell scripts to support command syntax standards (see intro(1)). This command parses positional parameters and checks for legal options. See getoptcvt(1) for usage and description.
hash [-r] [name ...]

For each name, the location in the search path of the command specified by name is determined and remembered by the shell. The -r option causes the shell to forget all remembered locations. If no arguments are given, information about remembered commands is presented. Hits is the number of times a command has been invoked by the shell process. Cost is a measure of the work required to locate a command in the search path. If a command is found in a "relative" directory in the search path, after changing to that directory, the stored location of that command is recalculated. Commands for which this will be done are indicated by an asterisk (*) adjacent to the hits information. Cost will be incremented when the recalculation is done.

jobs [-p|-l] [%jobid ...]
jobs -x command [arguments]

Reports all jobs that are stopped or executing in the background. If %jobid is omitted, all jobs that are stopped or running in the background will be reported. (See Job Control section below for more detail.)

kill [-sig] %job ...
kill -l

Sends either the TERM (terminate) signal or the specified signal to the specified jobs or processes. Signals are either given by number or by names (as given in signal(3HEAD) stripped of the prefix "SIG" with the exception that SIGCHD is named CHLD). If the signal being sent is TERM (terminate) or HUP (hangup), then the job or process will be sent a CONT (continue) signal if it is stopped. The argument job can be the process id of a process that is not a member of one of the active jobs. See Job Control section below for a description of the format of job. In the second form, kill -l, the signal numbers and names are listed. (See kill(1)).

login [argument ...]

Equivalent to `exec login argument...’. See login(1) for usage and description.

newgrp [argument]

Equivalent to exec newgrp argument. See newgrp(1) for usage and description.

pwd

Print the current working directory. See pwd(1) for usage and description.

read name ...

One line is read from the standard input and, using the internal field separator, IFS (normally space or tab), to delimit word boundaries, the first word is assigned to the first name, the second word to the second name, and so forth, with leftover words assigned to the last name. Lines can be continued using \newline. Characters other than newline can be quoted by preceding them with a backslash. These backslashes are removed before words are assigned to names, and no interpretation is done on the character that follows the backslash. The return code is 0, unless an EOF is encountered.
readonly [name...]  
The given names are marked readonly and the values of the these names may not be changed by subsequent assignment. If no arguments are given, a list of all readonly names is printed.

return [n]  
Causes a function to exit with the return value specified by n. If n is omitted, the return status is that of the last command executed.

set [-aeifhkntux [argument ...]]  
-a Mark variables which are modified or created for export.  
-e Exit immediately if a command exits with a non-zero exit status.  
-f Disable file name generation.  
-h Locate and remember function commands as functions are defined (function commands are normally located when the function is executed).  
-k All keyword arguments are placed in the environment for a command, not just those that precede the command name.  
-n Read commands but do not execute them.  
-t Exit after reading and executing one command.  
-u Treat unset variables as an error when substituting.  
-v Print shell input lines as they are read.  
-x Print commands and their arguments as they are executed.  
_ Do not change any of the flags; useful in setting $1 to _.

Using + rather than – causes these flags to be turned off. These flags can also be used upon invocation of the shell. The current set of flags may be found in $=. The remaining arguments are positional parameters and are assigned, in order, to $1, $2, ... If no arguments are given, the values of all names are printed.

shift [n]  
The positional parameters from $n+1... are renamed $1... If n is not given, it is assumed to be 1.

stop pid...  
Halt execution of the process number pid. (see ps(1)).

suspend  
Stops the execution of the current shell (but not if it is the login shell).

test  
Evaluate conditional expressions. See test(1) for usage and description.

times  
Print the accumulated user and system times for processes run from the shell.
The command `argument` is to be read and executed when the shell receives numeric or symbolic signal(s) (\(n\)). (Note: `argument` is scanned once when the trap is set and once when the trap is taken.) Trap commands are executed in order of signal number or corresponding symbolic names. Any attempt to set a trap on a signal that was ignored on entry to the current shell is ineffective. An attempt to trap on signal 11 (memory fault) produces an error. If `argument` is absent, all `trap(n)` are reset to their original values. If `argument` is the null string, this signal is ignored by the shell and by the commands it invokes. If \(n\) is 0, the command `argument` is executed on exit from the shell. The `trap` command with no arguments prints a list of commands associated with each signal number.

For each `name`, indicate how it would be interpreted if used as a command name.

- `ulimit` prints or sets hard or soft resource limits. These limits are described in `getrlimit(2)`.

If `limit` is not present, `ulimit` prints the specified limits. Any number of limits may be printed at one time. The `-a` option prints all limits.

If `limit` is present, `ulimit` sets the specified limit to `limit`. The string `unlimited` requests the largest valid limit. Limits may be set for only one resource at a time. Any user may set a soft limit to any value below the hard limit. Any user may lower a hard limit. Only a super-user may raise a hard limit. (See `su(1M)`.)

The `-H` option specifies a hard limit. The `-S` option specifies a soft limit. If neither option is specified, `ulimit` will set both limits and print the soft limit.

The following options specify the resource whose limits are to be printed or set. If no option is specified, the file size limit is printed or set.

- `-c` maximum core file size (in 512-byte blocks)
- `-d` maximum size of data segment or heap (in kbytes)
- `-f` maximum file size (in 512-byte blocks)
- `-n` maximum file descriptor plus 1
- `-s` maximum size of stack segment (in kbytes)
- `-t` maximum CPU time (in seconds)
- `-v` maximum size of virtual memory (in kbytes)

Run the `sysdef(1M)` command to obtain the maximum possible limits for your system. The values reported are in hexadecimal, but can be translated into decimal numbers using the `bc(1)` utility. See `swap(1M)`.)
As an example of `ulimit`, to limit the size of a core file dump to 0 Megabytes, type the following:

```
ulimit -c 0
```

`umask [nnn]`

The user file-creation mask is set to `nnn` (see `umask(1)`). If `nnn` is omitted, the current value of the mask is printed.

`unset [name ...]`

For each `name`, remove the corresponding variable or function value. The variables `PATH`, `PS1`, `PS2`, `MAILCHECK`, and `IFS` cannot be unset.

`wait [n]`

Wait for your background process whose process id is `n` and report its termination status. If `n` is omitted, all your shell’s currently active background processes are waited for and the return code will be zero.

**Invocation**

If the shell is invoked through `exec(2)` and the first character of argument zero is `-`, commands are initially read from `/etc/profile` and from `$HOME/.profile`, if such files exist. Thereafter, commands are read as described below, which is also the case when the shell is invoked as `/usr/bin/sh`. The flags below are interpreted by the shell on invocation only. *Note: Unless the `-c` or `-s` flag is specified, the first argument is assumed to be the name of a file containing commands, and the remaining arguments are passed as positional parameters to that command file:*

- `-c string`  
  If the `-c` flag is present commands are read from `string`.

- `-i`  
  If the `-i` flag is present or if the shell input and output are attached to a terminal, this shell is *interactive*. In this case, TERMINATE is ignored (so that `kill 0` does not kill an interactive shell) and INTERRUPT is caught and ignored (so that `wait` is interruptible). In all cases, QUIT is ignored by the shell.

- `-p`  
  If the `-p` flag is present, the shell will not set the effective user and group IDs to the real user and group IDs.

- `-r`  
  If the `-r` flag is present the shell is a restricted shell (see `rsh(1M)`).

- `-s`  
  If the `-s` flag is present or if no arguments remain, commands are read from the standard input. Any remaining arguments specify the positional parameters. Shell output (except for Special Commands) is written to file descriptor 2.

The remaining flags and arguments are described under the `set` command above.

**Job Control (jsh)**

When the shell is invoked as `jsh`, Job Control is enabled in addition to all of the functionality described previously for `sh`. Typically, Job Control is enabled for the interactive shell only. Non-interactive shells typically do not benefit from the added functionality of Job Control.
With Job Control enabled, every command or pipeline the user enters at the terminal is called a job. All jobs exist in one of the following states: foreground, background, or stopped. These terms are defined as follows:

1. A job in the foreground has read and write access to the controlling terminal.
2. A job in the background is denied read access and has conditional write access to the controlling terminal (see *stty*(1)).
3. A stopped job is a job that has been placed in a suspended state, usually as a result of a SIGTSTP signal (see *signal*(3HEAD)).

Every job that the shell starts is assigned a positive integer, called a job number which is tracked by the shell and will be used as an identifier to indicate a specific job. Additionally, the shell keeps track of the current and previous jobs. The current job is the most recent job to be started or restarted. The previous job is the first non-current job.

The acceptable syntax for a Job Identifier is of the form:

```
%jobid
```

where jobid may be specified in any of the following formats:

- % or +
  - For the current job.
- –
  - For the previous job.
- ?<string>
  - Specify the job for which the command line uniquely contains string.
- n
  - For job number n.
- pref
  - Where pref is a unique prefix of the command name. For example, if the command ls -l name were running in the background, it could be referred to as %ls. pref cannot contain blanks unless it is quoted.

When Job Control is enabled, the following commands are added to the user’s environment to manipulate jobs:

```
bg [ %jobid . . . ]
```

Resumes the execution of a stopped job in the background. If %jobid is omitted the current job is assumed.

```
fg [ %jobid . . . ]
```

Resumes the execution of a stopped job in the foreground, also moves an executing background job into the foreground. If %jobid is omitted the current job is assumed.

```
jobs [-p l -l] [ %jobid . . . ]
```

Reports all jobs that are stopped or executing in the background. If %jobid is omitted, all jobs that are stopped or running in the background will be reported. The following options will modify/enhance the output of jobs:
Report the process group ID and working directory of the jobs.

- Report only the process group ID of the jobs.

-x Replace any jobid found in command or arguments with the corresponding process group ID, and then execute command passing it arguments.

kill [-signal] %jobid
Built-in version of kill to provide the functionality of the kill command for processes identified with a jobid.

stop %jobid...
Stops the execution of a background job(s).

suspend
Stops the execution of the current shell (but not if it is the login shell).

wait [%jobid ...]
wait builtin accepts a job identifier. If %jobid is omitted wait behaves as described above under Special Commands.

Large File Behavior
See largefile(5) for the description of the behavior of sh and jsh when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXIT STATUS
Errors detected by the shell, such as syntax errors, cause the shell to return a non-zero exit status. If the shell is being used non-interactively execution of the shell file is abandoned. Otherwise, the shell returns the exit status of the last command executed (see also the exit command above).

jsh Only
If the shell is invoked as jsh and an attempt is made to exit the shell while there are stopped jobs, the shell issues one warning:

There are stopped jobs.

This is the only message. If another exit attempt is made, and there are still stopped jobs they will be sent a SIGHUP signal from the kernel and the shell is exited.

FILES
$HOME/.profile
/dev/null
/etc/profile
/tmp/sh*

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

<table>
<thead>
<tr>
<th>/usr/bin/jsh</th>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>
The use of setuid shell scripts is strongly discouraged. Words used for filenames in input/output redirection are not interpreted for filename generation (see File Name Generation section above). For example, cat file1 >a* will create a file named a*.

Because commands in pipelines are run as separate processes, variables set in a pipeline have no effect on the parent shell.

If you get the error message, “cannot fork, too many processes”, try using the wait(1) command to clean up your background processes. If this doesn’t help, the system process table is probably full or you have too many active foreground processes. There is a limit to the number of process ids associated with your login, and to the number the system can keep track of.

Only the last process in a pipeline can be waited for.

If a command is executed, and a command with the same name is installed in a directory in the search path before the directory where the original command was found, the shell will continue to exec the original command. Use the hash command to correct this situation.

The Bourne shell has a limitation on the effective UID for a process. If this UID is less than 100 (and not equal to the real UID of the process), then the UID is reset to the real UID of the process.

Because the shell implements both foreground and background jobs in the same process group, they all receive the same signals, which can lead to unexpected behavior. It is, therefore, recommended that other job control shells be used, especially in an interactive environment.
When the shell executes a shell script that attempts to execute a non-existent command interpreter, the shell returns an erroneous diagnostic message that the shell script file does not exist.
NAME | shell – run a command using shell
SYNOPSIS | shell command [command] ...
DESCRIPTION | The shell function concatenate its arguments, separating each by a space, and passes this string to the shell ($SHELL if set, otherwise /usr/bin/sh).
EXAMPLES | EXAMPLE 1 A sample output of shell command.
Since the Form and Menu Language does not directly support background processing, the shell function can be used instead.
'shell "build prog > /dev/null &"'
If you want the user to continue to be able to interact with the application while the background job is running, the output of an executable run by shell in the background must be redirected: to a file if you want to save the output, or to /dev/null if you don’t want to save it (or if there is no output), otherwise your application may appear to be hung until the background job finishes processing.
shell can also be used to execute a command that has the same name as an FMLI built-in function.
NOTES | The arguments to shell will be concatenate using spaces, which may or may not do what is expected. The variables set in local environments will not be expanded by the shell because "local" means "local to the current process."
ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>
SEE ALSO | sh(1), attributes(5)
The shell command interpreters `csh(1)`, `ksh(1)`, and `sh(1)` have special built-in commands. The commands `case`, `for`, `foreach`, `function`, `if`, `repeat`, `select`, `switch`, `until`, and `while` are commands in the syntax recognized by the shells. They are described in the Commands section of the manual pages of the respective shells. The remaining commands listed in the table below are built into the shells for reasons such as efficiency or data sharing between command invocations. They are described on their respective manual pages.

<table>
<thead>
<tr>
<th>Command</th>
<th>Shell</th>
</tr>
</thead>
<tbody>
<tr>
<td>alias</td>
<td>csh, ksh</td>
</tr>
<tr>
<td>bg</td>
<td>csh, ksh, sh</td>
</tr>
<tr>
<td>break</td>
<td>csh, ksh, sh</td>
</tr>
<tr>
<td>case</td>
<td>csh, ksh, sh</td>
</tr>
<tr>
<td>cd</td>
<td>csh, ksh, sh</td>
</tr>
<tr>
<td>chdir</td>
<td>csh, sh</td>
</tr>
<tr>
<td>continue</td>
<td>csh, ksh, sh</td>
</tr>
<tr>
<td>dirs</td>
<td>csh</td>
</tr>
<tr>
<td>echo</td>
<td>csh, ksh, sh</td>
</tr>
<tr>
<td>eval</td>
<td>csh, ksh, sh</td>
</tr>
<tr>
<td>exec</td>
<td>csh, ksh, sh</td>
</tr>
<tr>
<td>exit</td>
<td>csh, ksh, sh</td>
</tr>
<tr>
<td>export</td>
<td>ksh, sh</td>
</tr>
<tr>
<td>fc</td>
<td>ksh</td>
</tr>
<tr>
<td>fg</td>
<td>csh, ksh, sh</td>
</tr>
<tr>
<td>for</td>
<td>ksh, sh</td>
</tr>
<tr>
<td>foreach</td>
<td>csh</td>
</tr>
<tr>
<td>function</td>
<td>ksh</td>
</tr>
<tr>
<td>getopt</td>
<td>ksh</td>
</tr>
<tr>
<td>glob</td>
<td>ksh, sh</td>
</tr>
<tr>
<td>goto</td>
<td>csh</td>
</tr>
<tr>
<td>hash</td>
<td>ksh, sh</td>
</tr>
</tbody>
</table>
### Command Shell Builtins

<table>
<thead>
<tr>
<th>Command</th>
<th>Shell</th>
</tr>
</thead>
<tbody>
<tr>
<td>hashstat</td>
<td>csh</td>
</tr>
<tr>
<td>history</td>
<td>csh</td>
</tr>
<tr>
<td>if</td>
<td>csh, ksh, sh</td>
</tr>
<tr>
<td>jobs</td>
<td>csh, ksh, sh</td>
</tr>
<tr>
<td>kill</td>
<td>csh, ksh, sh</td>
</tr>
<tr>
<td>let</td>
<td>ksh</td>
</tr>
<tr>
<td>limit</td>
<td>csh</td>
</tr>
<tr>
<td>login</td>
<td>csh, ksh, sh</td>
</tr>
<tr>
<td>logout</td>
<td>csh, ksh, sh</td>
</tr>
<tr>
<td>nice</td>
<td>csh</td>
</tr>
<tr>
<td>newgrp</td>
<td>ksh, sh</td>
</tr>
<tr>
<td>notify</td>
<td>csh</td>
</tr>
<tr>
<td>onintr</td>
<td>csh</td>
</tr>
<tr>
<td>popd</td>
<td>csh</td>
</tr>
<tr>
<td>print</td>
<td>ksh</td>
</tr>
<tr>
<td>pushd</td>
<td>csh</td>
</tr>
<tr>
<td>pwd</td>
<td>ksh, sh</td>
</tr>
<tr>
<td>read</td>
<td>ksh, sh</td>
</tr>
<tr>
<td>readonly</td>
<td>ksh, sh</td>
</tr>
<tr>
<td>rehash</td>
<td>csh</td>
</tr>
<tr>
<td>repeat</td>
<td>csh</td>
</tr>
<tr>
<td>return</td>
<td>ksh, sh</td>
</tr>
<tr>
<td>select</td>
<td>ksh</td>
</tr>
<tr>
<td>set</td>
<td>csh, ksh, sh</td>
</tr>
<tr>
<td>setenv</td>
<td>csh</td>
</tr>
<tr>
<td>shift</td>
<td>csh, ksh, sh</td>
</tr>
<tr>
<td>source</td>
<td>csh</td>
</tr>
<tr>
<td>stop</td>
<td>csh, ksh, sh</td>
</tr>
<tr>
<td>suspend</td>
<td>csh, ksh, sh</td>
</tr>
<tr>
<td>Command</td>
<td>Shell</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>switch</td>
<td>csh</td>
</tr>
<tr>
<td>test</td>
<td>ksh, sh</td>
</tr>
<tr>
<td>time</td>
<td>csh</td>
</tr>
<tr>
<td>times</td>
<td>ksh, sh</td>
</tr>
<tr>
<td>trap</td>
<td>ksh, sh</td>
</tr>
<tr>
<td>type</td>
<td>ksh, sh</td>
</tr>
<tr>
<td>typeset</td>
<td>ksh</td>
</tr>
<tr>
<td>ulimit</td>
<td>ksh, sh</td>
</tr>
<tr>
<td>umask</td>
<td>csh, ksh, sh</td>
</tr>
<tr>
<td>unalias</td>
<td>csh, ksh</td>
</tr>
<tr>
<td>unhash</td>
<td>csh</td>
</tr>
<tr>
<td>unlimit</td>
<td>csh</td>
</tr>
<tr>
<td>unset</td>
<td>csh, ksh, sh</td>
</tr>
<tr>
<td>unsetenv</td>
<td>csh</td>
</tr>
<tr>
<td>until</td>
<td>ksh, sh</td>
</tr>
<tr>
<td>wait</td>
<td>csh, ksh, sh</td>
</tr>
<tr>
<td>whence</td>
<td>ksh</td>
</tr>
<tr>
<td>while</td>
<td>csh, ksh, sh</td>
</tr>
</tbody>
</table>

**Bourne Shell, sh, Special Commands**

Input/output redirection is now permitted for these commands. File descriptor 1 is the default output location. When Job Control is enabled, additional *Special Commands* are added to the shell’s environment.

In addition to these built-in reserved command words, *sh* also uses:

- **:** No effect; the command does nothing. A zero exit code is returned.

- **.filename** Read and execute commands from *filename* and return. The search path specified by *PATH* is used to find the directory containing *filename*.

**C shell, csh**

Built-in commands are executed within the C shell. If a built-in command occurs as any component of a pipeline except the last, it is executed in a subshell. In addition to these built-in reserved command words, *csh* also uses:

- **:** Null command. This command is interpreted, but performs no action.
Input/Output redirection is permitted. Unless otherwise indicated, the output is written on file descriptor 1 and the exit status, when there is no syntax error, is zero.

Commands that are preceded by one or two * (asterisks) are treated specially in the following ways:

1. Variable assignment lists preceding the command remain in effect when the command completes.
2. I/O redirections are processed after variable assignments.
3. Errors cause a script that contains them to abort.
4. Words, following a command preceded by ** that are in the format of a variable assignment, are expanded with the same rules as a variable assignment. This means that tilde substitution is performed after the = sign and word splitting and file name generation are not performed.

In addition to these built-in reserved command words, ksh also uses:

* : [ arg ... ] The command only expands parameters.
* . file [ arg ... ] Read the complete file then execute the commands. The commands are executed in the current shell environment. The search path specified by PATH is used to find the directory containing file. If any arguments arg are given, they become the positional parameters. Otherwise, the positional parameters are unchanged. The exit status is the exit status of the last command executed. the loop termination test.

**SEE ALSO**

Korn Shell, ksh, Special Commands

intro(1), alias(1), break(1), cd(1), chmod(1), csh(1), echo(1), exec(1), exit(1), find(1), getoptcvt(1), getopts(1), glob(1), hash(1), history(1), jobs(1), kill(1), ksh(1), let(1), limit(1), login(1), logout(1), newgrp(1), nice(1), nohup(1), print(1), pwd(1), read(1), readonly(1), set(1), sh(1), shift(1), suspend(1), test(1B), time(1), times(1), trap(1), typeset(1), umask(1), wait(1), chdir(2), chmod(2), creat(2), umask(2), getopt(3C), profile(4), environ(5)

shell_builtins(1)
shift(1)

NAME  

shift – shell built-in function to traverse either a shell’s argument list or a list of field-separated words

sh  shift [n]
csh  shift [variable]
ksh  * shift [n]

sh  The positional parameters from $n+1 \ldots$ are renamed $1 \ldots$. If n is not given, it is assumed to be 1.
csh  The components of argv, or variable, if supplied, are shifted to the left, discarding the first component. It is an error for the variable not to be set or to have a null value.
ksh  The positional parameters from $n+1 \ldots$ are renamed $1 \ldots$, default $n$ is 1. The parameter $n$ can be any arithmetic expression that evaluates to a non-negative number less than or equal to $\#$.

On this man page, ksh(1) commands that are preceded by one or two * (asterisks) are treated specially in the following ways:

1. Variable assignment lists preceding the command remain in effect when the command completes.
2. I/O redirections are processed after variable assignments.
3. Errors cause a script that contains them to abort.
4. Words, following a command preceded by ** that are in the format of a variable assignment, are expanded with the same rules as a variable assignment. This means that tilde substitution is performed after the = sign and word splitting and file name generation are not performed.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO  csh(1), ksh(1), sh(1), attributes(5)
shutdown – close down the system at a given time

SYNOPSIS
/usr/ucb/shutdown [-fhknr] time [warning-message...]

DESCRIPTION
shutdown provides an automated procedure to notify users when the system is to be shut down. time specifies when shutdown will bring the system down; it may be the word now (indicating an immediate shutdown), or it may specify a future time in one of two formats: +number and hour:min. The first form brings the system down in number minutes, and the second brings the system down at the time of day indicated in 24-hour notation.

At intervals that get closer as the apocalypse approaches, warning messages are displayed at terminals of all logged-in users, and of users who have remote mounts on that machine.

At shutdown time a message is written to the system log daemon, syslogd(1M), containing the time of shutdown, the instigator of the shutdown, and the reason. Then a terminate signal is sent to init, which brings the system down to single-user mode.

OPTIONS
As an alternative to the above procedure, these options can be specified:

- f Arrange, in the manner of fastboot(1B), that when the system is rebooted, the file systems will not be checked.
- h Execute halt(1M).
- k Simulate shutdown of the system. Do not actually shut down the system.
- n Prevent the normal sync(2) before stopping.
- r Execute reboot(1M).

FILES
/etc/rmtab remote mounted file system table

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

SEE ALSO
fastboot(1B), login(1), halt(1M), reboot(1M), syslogd(1M), sync(2), rmtab(4), attributes(5)

NOTES
Only allows you to bring the system down between now and 23:59 if you use the absolute time for shutdown.
size(1)

NAME  size – print section sizes in bytes of object files

SYNOPSIS  size [-f] [-F] [-n] [-o] [-V] [-x] filename...

DESCRIPTION  The size command produces segment or section size information in bytes for each loaded section in ELF object files. size prints out the size of the text, data, and bss (uninitialized data) segments (or sections) and their total.

size processes ELF object files entered on the command line. If an archive file is input to the size command, the information for each object file in the archive is displayed.

When calculating segment information, the size command prints out the total file size of the non-writable segments, the total file size of the writable segments, and the total memory size of the writable segments minus the total file size of the writable segments.

If it cannot calculate segment information, size calculates section information. When calculating section information, it prints out the total size of sections that are allocatable, non-writable, and not NOBITS, the total size of the sections that are allocatable, writable, and not NOBITS, and the total size of the writable sections of type NOBITS. NOBITS sections do not actually take up space in the filename.

If size cannot calculate either segment or section information, it prints an error message and stops processing the file.

OPTIONS  The following options are supported:

-f  Prints out the size of each allocatable section, the name of the section, and the total of the section sizes. If there is no section data, size prints out an error message and stops processing the file.

-F  Prints out the size of each loadable segment, the permission flags of the segment, then the total of the loadable segment sizes. If there is no segment data, size prints an error message and stops processing the file.

-n  Prints out non-loadable segment or non-allocatable section sizes. If segment data exists, size prints out the memory size of each loadable segment or file size of each non-loadable segment, the permission flags, and the total size of the segments. If there is no segment data, size prints out, for each allocatable and non-allocatable section, the memory size, the section name, and the total size of the sections. If there is no segment or section data, size prints an error message and stops processing.

-o  Prints numbers in octal, not decimal.

-V  Prints the version information for the size command on the standard error output.

-x  Prints numbers in hexadecimal, not decimal.

EXAMPLES  The examples below are typical size output.
EXAMPLE 1

example size filename
2724 + 88 + 0 = 2812

EXAMPLE 2

example size -f filename
26(.text) + 5(.init) + 5(.fini) = 36

EXAMPLE 3

example size -F filename
2724(r-x) + 88(rwx) + 0(rwx) = 2812 ... (If statically linked)

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWbtool</td>
</tr>
</tbody>
</table>

SEE ALSO

as(1), cc(1B), ld(1), ar(3HEAD), a.out(4), attributes(5)

NOTES

Since the size of bss sections is not known until link-edit time, the size command will not give the true total size of pre-linked objects.
sleep(1)

NAME
sleep – suspend execution for an interval

SYNOPSIS
sleep time

DESCRIPTION
The sleep utility will suspend execution for at least the integral number of seconds
specified by the time operand.

OPERANDS
The following operands are supported:

  time     A non-negative decimal integer specifying the number of seconds for
           which to suspend execution.

EXAMPLES
EXAMPLE 1 Example of the sleep command.

To execute a command after a certain amount of time:

(sleep 105; command) &

or to execute a command every so often:

while true
  do
    command
    sleep 37
  done

ENVIRONMENT
See environ(5) for descriptions of the following environment variables that affect the
execution of sleep: LC_CTYPE, LC_MESSAGES, and NLSPATH.

VARIABLES

EXIT STATUS
The following exit values are returned:

  0      The execution was successfully suspended for at least time seconds, or a
         SIGALRM signal was received (see NOTES).

  >0     An error has occurred.

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

+-----------------+-----------------+
| ATTRIBUTE TYPE   | ATTRIBUTE VALUE |
+-----------------+-----------------+
| Availability    | SUNWcsu         |
+-----------------+-----------------+

SEE ALSO
wait(1), alarm(2), sleep(3C), wait(3UCB), attributes(5), environ(5)

NOTES
If the sleep utility receives a SIGALRM signal, one of the following actions will be
taken:

- Terminate normally with a zero exit status.
- Effectively ignore the signal.

The sleep utility will take the standard action for all other signals.
**NAME**
smart2cfg – Compaq Smart-2 EISA/PCI and Smart-2SL PCI Array Controller ioctl utility

**SYNOPSIS**
smart2cfg -c [controller_num]
smart2cfg -d [controller_num]
smart2cfg -h
smart2cfg -l logical_drive_num [controller_num]
smart2cfg -p physical_drive_num bus_num [controller_num]

**DESCRIPTION**
The smart2cfg utility issues controller-specific ioctls to the Compaq Smart-2 EISA/PCI and Smart-2SL PCI array controller using the smartii(7D) driver.

smart2cfg provides information about the Smart-2 and Smart-2SL controllers installed on the system, the Logical and Physical drives as well as the details of the ReadWrite cache present on each controller. The utility is text based and is driven by command line arguments. smart2cfg and the smartii(7D) driver communicate using ioctls. smart2cfg also supports multiple commands.

**OPTIONS**
The following options are supported:

- `-c controller_num`
  Prints cache details of the cache on controller controller_num.

- `-d controller_num`
  Prints details of all the physical disks, all the logical drives, and the cache on controller controller_num.

- `-h`
  Provides on-line help for the smart2cfg utility.

- `-l logical_drive_num controller_num`
  Prints logical drive details of the drive logical_drive_num on controller controller_num.

- `-p physical_drive_num bus_num controller_num`
  Prints physical drive details of the disk physical_drive_num on bus bus_num on controller controller_num.

**EXAMPLES**

**EXAMPLE 1** Providing details of the physical disk
To provide details of the physical disk with SCSI ID 0, on Bus 0, on controller 0:

eample% smart2cfg -p -0 0 0

**EXAMPLE 2** Providing logical drive details
To provide logical drive details of logical drive 0 on controller 0:

eample% smart2cfg -l 0 0
EXAMPLE 3 Providing cache details

To provide cache details of controller 0:

```
example% smart2cfg -c 0
```

EXAMPLE 4 Providing information of all physical disks, logical drives, and cache on a controller

To provide information on all physical disks, logical drives, and cache on controller 0:

```
smart2cfg -d
```

EXAMPLE 5 Providing details of the disk

To provide details of the disk with SCSI ID 0, on Bus 0, on controller 0, and logical drive details of logical drive 0 on controller 1:

```
example% smart2cfg -p 0 0 0 -l 0 1
```

FILES

```
/devices/eisa/smartii<>/instance>,<ioaddr>:ioctl
/devices/pci@0,<bus_num>/pci1014,22<device_num>/pci11,4030@0:ioctl
/devices/pci@0,<bus_num>/pci1014,22<device_num>/pci11,4031@0:ioctl
```

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>IA</td>
</tr>
</tbody>
</table>

SEE ALSO

attributes(5), smartii(7D)

NOTES

If the controller is not specified, the first controller is taken as default.
soelim – resolve and eliminate .so requests from nroff or troff input

soelim reads the specified files or the standard input and performs the textual inclusion implied by the \texttt{nroff(1)} directives of the form

\texttt{.so} \texttt{somefile}

when they appear at the beginning of input lines. This is useful since programs such as \texttt{tbl(1)} do not normally do this; it allows the placement of individual tables in separate files to be run as a part of a large document.

An argument consisting of `−' is taken to be a file name corresponding to the standard input.

Note: Inclusion can be suppressed by using `' ' instead of `.', that is,

` so /usr/share/lib/tmac/tmac.s

**EXAMPLE 1** A sample of the soelim command.

A sample usage of soelim would be

```
example$ soelim exum?.n | tbl | nroff -ms | col | lpr
```

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWdoc</td>
</tr>
</tbody>
</table>

**SEE ALSO** more(1), nroff(1), tbl(1), attributes(5)
solregis(1)

NAME  solregis – Solaris user registration

SYNOPSIS  /usr/dt/bin/solregis [-dc]

DESCRIPTION  The solregis command initiates the Solaris user registration procedure. This allows users to register with Sun Microsystems and receive information about Solaris. Normally, solregis is executed in conditional mode as a part of desktop login so that users are prompted at desktop start up time to register, unless they have already done so.

OPTIONS  The following options are supported:

- d  Delay display of the initial screen until a window manager has asserted control of the X display.

- c  Conditional mode. If specified, solregis will exit without any dialog displayed if: (1) $HOME/.solregis/disable exists, (2) DISABLE=1 is specified in /etc/default/solregis, or (3) the user has already registered.

USAGE  The following resources can control the behavior and appearance of solregis:

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Value Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>disable</td>
<td>Disable</td>
<td>Boolean</td>
<td>False</td>
</tr>
<tr>
<td>localeChoices</td>
<td>LocaleChoices</td>
<td>Int</td>
<td>1</td>
</tr>
<tr>
<td>action0</td>
<td>Action</td>
<td>String</td>
<td>/usr/dt/bin/hotjava</td>
</tr>
<tr>
<td>initialURL0</td>
<td>URL</td>
<td>String</td>
<td>file:///usr/dt/app-config/solregis/EReg.html</td>
</tr>
<tr>
<td>localeChoice0</td>
<td>LocaleChoice</td>
<td>String</td>
<td>null</td>
</tr>
<tr>
<td>actionn</td>
<td>Action</td>
<td>String</td>
<td>null for n&gt;0</td>
</tr>
<tr>
<td>initialURLn</td>
<td>URL</td>
<td>String</td>
<td>null for n&gt;0</td>
</tr>
<tr>
<td>printContext</td>
<td>PrintContext</td>
<td>String</td>
<td>thisorgunit</td>
</tr>
</tbody>
</table>

disable  If TRUE, when executed in conditional mode solregis simply exits without displaying anything.

localeChoices  Specifies the number of localeChoice, actionn and initialURLn sets. The first set is 0, so if localeChoices is 1, localeChoice0, action0, and initialURL0 are the only active resources. If localeChoices is 1, none of the localeChoice strings are displayed, and action0, and so forth. are
used. If localeChoices is greater than 1, each
localeChoice string is made an element in an
exclusive choice list and the index of the selected item
controls which action and initialURL resources
are applied.
localeChoice specifies the string presented to the user for this choice.
action specifies the file name of the command to be executed
(normally expected to be a World Wide Web browser)
when the user selects "Register Now", or the special
string "print". If "print" is specified, the initialURL string must be a file name on the local system, naming
a file which is to be printed after prompting the user
for a print destination.
initialURL specifies the argument to be passed to action for
initial registration. This will normally be the Universal
Resource Locator for the initial page to be displayed by
the World Wide Web browser.
printContext XFN naming context under which the printers to
display to the user if the special "print" action are
named, in the service/printer context. For example, if
the default printContext "thisorgunit" is used, the
printers in thisorgunit/service/printer are
displayed.

ENVIRONMENT VARIABLES
See environ(5) for descriptions of the following environment variables that affect the
execution of solregis: HOME, LANG, LC_MESSAGES, and NLSPATH.

EXIT STATUS
The following exit values are returned:
0 Successful completion.
>0 An error occurred.

FILES
/etc/default/solregis
Default values.
/$HOME/.solregis/uprops
User registration information.
/$HOME/.solregis/disable
Users disabled from registration.
/usr/dt/app-defaults/C/Solregis
Default locale resources.
/usr/dt/app-defaults/$LANG/Solregis
Default localized resources.
solregis(1)

/etc/dt/app-defaults/C/Solregis
  Default installation resources.
/usr/dt/app-defaults/$LANG/Solregis
  Localized installation resources.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNW*sregu</td>
</tr>
</tbody>
</table>

SEE ALSO
attributes(5), environ(5)
sort(1)

NAME
sort – sort, merge, or sequence check text files

SYNOPSIS
/usr/bin/sort [-bcdfimNru] [-k keydef] [-o output] [-S kmem]
[-t char] [-T directory] [-y [kmem]] [-z recsz] [+pos1 [-pos2]] [file...]

/usr/xpg4/bin/sort [-bcdfimNru] [-k keydef] [-o output] [-S kmem]
[-t char] [-T directory] [-y [kmem]] [-z recsz] [+pos1 [-pos2]] [file...]

DESCRIPTION
The sort command sorts lines of all the named files together and writes the result on
the standard output.

Comparisons are based on one or more sort keys extracted from each line of input. By
default, there is one sort key, the entire input line. Lines are ordered according to the
collating sequence of the current locale.

OPTIONS
The following options alter the default behavior:

/usr/bin/sort
-c Checks that the single input file is ordered as specified by the
arguments and the collating sequence of the current locale. The
exit code is set and no output is produced unless the file is out of
sort.

/usr/xpg4/bin/sort
-c Same as /usr/bin/sort except no output is produced under any
circumstances.

-m Merges only. The input files are assumed to be already sorted.

-o output Specifies the name of an output file to be used instead of the
standard output. This file can be the same as one of the input files.

-S kmem Specifies the maximum amount of swap-based memory used for
sorting, in kilobytes (the default unit). kmem can also be specified
directly as a number of bytes (b), kilobytes (k), megabytes (m),
gigabytes (g), or terabytes (t); or as a percentage (%) of the
installed physical memory.

-T directory Specifies the directory in which to place temporary files.

-u Unique: suppresses all but one in each set of lines having equal
keys. If used with the -c option, checks that there are no lines with
duplicate keys in addition to checking that the input file is sorted.

-y kmem (obsolete). This option was used to specify the amount of main
memory initially used by sort. Its functionality is not appropriate
for a virtual memory system; memory usage for sort is now
specified using the -S option.

-z recsz (obsolete). This option was used to prevent abnormal termination
when lines longer than the system-dependent default buffer size
are encountered. Because sort automatically allocates buffers
large enough to hold the longest line, this option has no effect.
The default sort order depends on the value of \texttt{LC_COLLATE}. If \texttt{LC_COLLATE} is set to \texttt{C}, sorting will be in ASCII order. If \texttt{LC_COLLATE} is set to \texttt{en\_US}, sorting is case insensitive except when the two strings are otherwise equal and one has an uppercase letter earlier than the other. Other locales will have other sort orders.

The following options override the default ordering rules. When ordering options appear independent of any key field specifications, the requested field ordering rules are applied globally to all sort keys. When attached to a specific key (see Sort Key Options), the specified ordering options override all global ordering options for that key. In the obsolescent forms, if one or more of these options follows a \texttt{+pos1} option, it will affect only the key field specified by that preceding option.

- \textbf{\texttt{-d}} \quad “Dictionary’’ order: only letters, digits, and blanks (spaces and tabs) are significant in comparisons.

- \textbf{\texttt{-f}} \quad Folds lower-case letters into upper case.

- \textbf{\texttt{-i}} \quad Ignores non-printable characters.

- \textbf{\texttt{-M}} \quad Compares as months. The first three non-blank characters of the field are folded to upper case and compared. For example, in English the sorting order is “JAN” < “FEB” < \ldots < “DEC”. Invalid fields compare low to “JAN”. The \texttt{-M} option implies the \texttt{-b} option (see below).

- \textbf{\texttt{-n}} \quad Restricts the sort key to an initial numeric string, consisting of optional blank characters, optional minus sign, and zero or more digits with an optional radix character and thousands separators (as defined in the current locale), which will be sorted by arithmetic value. An empty digit string is treated as zero. Leading zeros and signs on zeros do not affect ordering.

- \textbf{\texttt{-r}} \quad Reverses the sense of comparisons.

### Field Separator Options

The treatment of field separators can be altered using the following options:

- \textbf{\texttt{-b}} \quad Ignores leading blank characters when determining the starting and ending positions of a restricted sort key. If the \texttt{-b} option is specified before the first sort key option, it is applied to all sort key options. Otherwise, the \texttt{-b} option can be attached independently to each \texttt{-k field\_start, field\_end}, or \texttt{+pos1} or \texttt{-pos2} option-argument (see below).

- \textbf{\texttt{-t char}} \quad Use \texttt{char} as the field separator character. \texttt{char} is not considered to be part of a field (although it can be included in a sort key). Each occurrence of \texttt{char} is significant (for example, \texttt{<char><char>} delimits an empty field). If \texttt{-t} is not specified, blank characters are used as default field separators; each maximal non-empty sequence of blank characters that follows a non-blank character is a field separator.
Sort keys can be specified using the options:

- **-k keydef**

  The `keydef` argument is a restricted sort key field definition. The format of this definition is:

  ```
  -k field_start [type] [.field_end [type]]
  ```

  where:

  - `field_start` and `field_end`
    define a key field restricted to a portion of the line.

  - `type`
    is a modifier from the list of characters bdfimnr. The `b` modifier behaves like the `-b` option, but applies only to the `field_start` or `field_end` to which it is attached and characters within a field are counted from the first non-blank character in the field. (This applies separately to `first_character` and `last_character`.) The other modifiers behave like the corresponding options, but apply only to the key field to which they are attached. They have this effect if specified with `field_start`, `field_end` or both. If any modifier is attached to a `field_start` or to a `field_end`, no option applies to either.

When there are multiple key fields, later keys are compared only after all earlier keys compare equal. Except when the `-u` option is specified, lines that otherwise compare equal are ordered as if none of the options `-d`, `-e`, `-i`, `-n` or `-k` were present (but with `-r` still in effect, if it was specified) and with all bytes in the lines significant to the comparison.

The notation:

```
-k field_start [type] [.field_end [type]]
```

defines a key field that begins at `field_start` and ends at `field_end` inclusive, unless `field_start` falls beyond the end of the line or after `field_end`, in which case the key field is empty. A missing `field_end` means the last character of the line.

A field comprises a maximal sequence of non-separating characters and, in the absence of option `-t`, any preceding field separator.

The `field_start` portion of the `keydef` option-argument has the form:

```
field_number [.first_character]
```

Fields and characters within fields are numbered starting with 1. `field_number` and `first_character`, interpreted as positive decimal integers, specify the first character to be used as part of a sort key. If `.first_character` is omitted, it refers to the first character of the field.
The field_end portion of the keydef option-argument has the form:

\[ \text{field\_number [.last\_character]} \]

The field_number is as described above for field_start. last_character, interpreted as a non-negative decimal integer, specifies the last character to be used as part of the sort key. If last_character evaluates to zero or .last_character is omitted, it refers to the last character of the field specified by field_number.

If the -b option or b type modifier is in effect, characters within a field are counted from the first non-blank character in the field. (This applies separately to first_character and last_character.)

[pos1 [-pos2]] (obsolete). Provide functionality equivalent to the -kkeydef option.

pos1 and pos2 each have the form m.n optionally followed by one or more of the flags bdfiMnr. A starting position specified by +m.n is interpreted to mean the n+1st character in the m+1st field. A missing .n means .0, indicating the first character of the m+1st field. If the b flag is in effect n is counted from the first non-blank in the m+1st field; +m.0b refers to the first non-blank character in the m+1st field.

A last position specified by −m.n is interpreted to mean the nth character (including separators) after the last character of the mth field. A missing .n means .0, indicating the last character of the mth field. If the b flag is in effect n is counted from the last leading blank in the m+1st field; −m.1b refers to the first non-blank in the m+1st field.

The fully specified +posI −pos2 form with type modifiers T and U:

\[ +w.xT -y.zU \]

is equivalent to:

\[ \text{undefined} \quad (z==0 \, \& \, U \, \text{contains } b \, \& \, -t \, \text{is present}) \]
\[ -k \, w+1.x+1T,y.0U \quad (z==0 \, \text{otherwise}) \]
\[ -k \, w+1.x+1T,y+1.zU \quad (z > 0) \]

Implementations support at least nine occurrences of the sort keys (the -k option and obsolescent +pos1 and −pos2) which are significant in command line order. If no sort key is specified, a default sort key of the entire line is used.

OPERANDS

The following operand is supported:

\[ \text{file} \]

A path name of a file to be sorted, merged or checked. If no file operands are specified, or if a file operand is −, the standard input will be used.
 USAGE

See `largefile(5)` for the description of the behavior of `sort` when encountering files greater than or equal to 2 Gbyte \((2^{31})\) bytes.

 EXAMPLES

In the following examples, first the preferred and then the obsolete way of specifying `sort` keys are given as an aid to understanding the relationship between the two forms.

 **EXAMPLE 1** Sorting with the second field as a sort key

Either of the following commands sorts the contents of `infile` with the second field as the sort key:

```
example\% sort -k 2,2 infile
example\% sort +1 -2 infile
```

 **EXAMPLE 2** Sorting in reverse order

Either of the following commands sorts, in reverse order, the contents of `infile1` and `infile2`, placing the output in `outfile` and using the second character of the second field as the sort key (assuming that the first character of the second field is the field separator):

```
example\% sort -r -o outfile -k 2.2,2.2 infile1 infile2
example\% sort -r -o outfile +1.1 -1.2 infile1 infile2
```

 **EXAMPLE 3** Sorting using a specified character in one of the files

Either of the following commands sorts the contents of `infile1` and `infile2` using the second non-blank character of the second field as the sort key:

```
example\% sort -k 2.2b,2.2b infile1 infile2
example\% sort +1.1b -1.2b infile1 infile2
```

 **EXAMPLE 4** Sorting by numeric user ID

Either of the following commands prints the `passwd(4)` file (user database) sorted by the numeric user ID (the third colon-separated field):

```
example\% sort -t : -k 3,3n /etc/passwd
example\% sort -t : +2 -3n /etc/passwd
```

 **EXAMPLE 5** Printing sorted lines excluding lines that duplicate a field

Either of the following commands prints the lines of the already sorted file `infile`, suppressing all but one occurrence of lines having the same third field:

```
example\% sort -um -k 3.1,3.0 infile
example\% sort -um +2.0 -3.0 infile
```
EXAMPLE 6 Sorting by host IP address

Either of the following commands prints the hosts(4) file (IPv4 hosts database), sorted by the numeric IP address (the first four numeric fields):

```bash
example$ sort -t . -k 1,1n -k 2,2n -k 3,3n -k 4,4n /etc/hosts
example$ sort -t . +0 -1n +1 -2n +2 -3n +3 -4n /etc/hosts
```

Since ‘.’ is both the field delimiter and, in many locales, the decimal separator, failure to specify both ends of the field will lead to results where the second field is interpreted as a fractional portion of the first, and so forth.

**ENVIRONMENT VARIABLES**

See environ(5) for descriptions of the following environment variables that affect the execution of sort: LC_COLLATE, LC_MESSAGES, and NLSPATH.

- **LC_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single- versus multi-byte characters in arguments and input files) and the behavior of character classification for the -b, -d, -f, -i and -n options.
- **LC_NUMERIC** Determine the locale for the definition of the radix character and thousands separator for the -n option.

**EXIT STATUS**

The following exit values are returned:

- **0** All input files were output successfully, or -c was specified and the input file was correctly sorted.
- **1** Under the -c option, the file was not ordered as specified, or if the -c and -u options were both specified, two input lines were found with equal keys.
- **>1** An error occurred.

**FILES**

- `/var/tmp/stm???
  temporary files`

**ATTRIBUTES**

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

```
<table>
<thead>
<tr>
<th>/usr/bin/sort</th>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
<td></td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>/usr/xpg4/bin/sort</th>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWxcu4</td>
<td></td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
<td></td>
</tr>
</tbody>
</table>
```
Comments and exits with non-zero status for various trouble conditions (for example, when input lines are too long), and for disorders discovered under the -c option.

When the last line of an input file is missing a new-line character, sort appends one, prints a warning message, and continues.

sort does not guarantee preservation of relative line ordering on equal keys.

One can tune sort performance for a specific scenario using the -S option. However, one should note in particular that sort has greater knowledge of how to use a finite amount of memory for sorting than the virtual memory system. Thus, a sort invoked to request an extremely large amount of memory via the -S option could perform extremely poorly.

As noted, certain of the field modifiers (such as -M and -d) cause the interpretation of input data to be done with reference to locale-specific settings. The results of this interpretation can be unexpected if one’s expectations are not aligned with the conventions established by the locale. In the case of the month keys, sort does not attempt to compensate for "approximate" month abbreviations. The precise month abbreviations from nl_langinfo(3C) or strftime(3C) are the only ones recognized.

For printable or dictionary order, if these concepts are not well-defined by the locale, an empty sort key may be the result, leading to the next key being the significant one for determining the appropriate ordering.

SEE ALSO
comm(1), join(1), uniq(1), nl_langinfo(3C), strftime(3C), hosts(4), passwd(4), attributes(5), environ(5), largefile(5), XPG4(5)

DIAGNOSTICS
Comments and exits with non-zero status for various trouble conditions (for example, when input lines are too long), and for disorders discovered under the -c option.

NOTES
When the last line of an input file is missing a new-line character, sort appends one, prints a warning message, and continues.

sort does not guarantee preservation of relative line ordering on equal keys.

One can tune sort performance for a specific scenario using the -S option. However, one should note in particular that sort has greater knowledge of how to use a finite amount of memory for sorting than the virtual memory system. Thus, a sort invoked to request an extremely large amount of memory via the -S option could perform extremely poorly.

As noted, certain of the field modifiers (such as -M and -d) cause the interpretation of input data to be done with reference to locale-specific settings. The results of this interpretation can be unexpected if one’s expectations are not aligned with the conventions established by the locale. In the case of the month keys, sort does not attempt to compensate for "approximate" month abbreviations. The precise month abbreviations from nl_langinfo(3C) or strftime(3C) are the only ones recognized. For printable or dictionary order, if these concepts are not well-defined by the locale, an empty sort key may be the result, leading to the next key being the significant one for determining the appropriate ordering.
sortbib(1)

NAME  sortbib – sort a bibliographic database

SYNOPSIS  sortbib [-s KEYS] database...

DESCRIPTION  sortbib sorts files of records containing refer key-letters by user-specified keys. Records may be separated by blank lines, or by ‘.’ and ‘.’’ delimiters, but the two styles may not be mixed together. This program reads through each database and pulls out key fields, which are sorted separately. The sorted key fields contain the file pointer, byte offset, and length of corresponding records. These records are delivered using disk seeks and reads, so sortbib may not be used in a pipeline to read standard input.

The most common key-letters and their meanings are given below.

%A  Author’s name
%B  Book containing article referenced
%C  City (place of publication)
%D  Date of publication
%E  Editor of book containing article referenced
%F  Footnote number or label (supplied by refer)
%G  Government order number
%H  Header commentary, printed before reference
%I  Issuer (publisher)
%J  Journal containing article
%K  Keywords to use in locating reference
%L  Label field used by -k option of refer
%M  Bell Labs Memorandum (undefined)
%N  Number within volume
%O  Other commentary, printed at end of reference
%P  Page number(s)
%Q  Corporate or Foreign Author (unreversed)
%R  Report, paper, or thesis (unpublished)
%S  Series title
%T  Title of article or book
%V  Volume number
%X  Abstract — used by roffbib, not by refer
%Y, Z  Ignored by refer
By default, `sortbib` alphabetizes by the first `%A` and the `%D` fields, which contain the senior author and date.

`sortbib` sorts on the last word on the `%A` line, which is assumed to be the author’s last name. A word in the final position, such as ‘jr.’ or ‘ed.’, will be ignored if the name beforehand ends with a comma. Authors with two-word last names or unusual constructions can be sorted correctly by using the `nroff` convention ‘\0’ in place of a blank. A `%Q` field is considered to be the same as `%A`, except sorting begins with the first, not the last, word. `sortbib` sorts on the last word of the `%D` line, usually the year. It also ignores leading articles (like ‘A’ or ‘The’) when sorting by titles in the `%T` or `%J` fields; it will ignore articles of any modern European language. If a sort-significant field is absent from a record, `sortbib` places that record before other records containing that field.

No more than 16 databases may be sorted together at one time. Records longer than 4096 characters will be truncated.

**OPTIONS**

- `-sKEYS` Specify new `KEYS`. For instance, `-sATD` will sort by author, title, and date, while `-sA+D` will sort by all authors, and date. Sort keys past the fourth are not meaningful.

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWdoc</td>
</tr>
</tbody>
</table>

**SEE ALSO**

`addbib(1)`, `indexbib(1)`, `lookbib(1)`, `refer(1)`, `roffbib(1)`, `attributes(5)`

**BUGS**

Records with missing author fields should probably be sorted by title.
sotruss(1)

NAME
sotruss – trace shared library procedure calls

SYNOPSIS
/usr/bin/sotruss [-f] [-F bindfromlist] [-T bindtolist] [-o outputfile]
            executable [executable arguments...]

DESCRIPTION
sotruss executes the specified command and produces a trace of the library calls that
it performs. Each line of the trace output reports what bindings are occurring between
dynamic objects as each procedure call is executed. sotruss traces all of the
procedure calls that occur between dynamic objects via the Procedure Linkage Table, so
only those procedure calls which are bound via the Procedure Linkage Table will be
traced. See Linker and Libraries Guide

OPTIONS
- F bindfromlist A colon-separated list of libraries that are to be traced.
Only calls from these libraries will be traced. The
default is to trace calls from the main executable only.

- T bindtolist A colon-separated list of libraries that are to be traced.
Only calls to these libraries will be traced. The default
is to trace all calls.

- o outputfile sotruss output will be directed to the
output file. If this option is combined with the -f option then the pid
of the executing program will be placed at the end of the
filename. By default sotruss output is placed on
stderr.

- f Follow all children created by fork() and print
truss output on each child process. This option will
also cause a pid to be output on each truss output
line.

EXAMPLES
EXAMPLE 1 An example of sotruss.

A simple example shows the tracing of a simple ls command:

$ sotruss ls | more

ls -> libc.so.1:*atexit(0xef7d7d1c, 0x23c00, 0x0)
ls -> libc.so.1:*atexit(0x1392c, 0xef7d7d1c, 0xef621bb0)
ls -> libc.so.1:*setlocale(0x6, 0x1396c, 0xef621ba8)
ls -> libc.so.1:*textdomain(0x13970, 0x1396c, 0xef621ba8)
ls -> libc.so.1:*time(0x0, 0xef61f6fc, 0xef621ba8)
ls -> libc.so.1:*isatty(0x1, 0xef61f6fc, 0x0)
ls -> libc.so.1:*getopt(0x1, 0xef6ff88fc, 0x13980)
ls -> libc.so.1:*malloc(0x1000, 0x0, 0x0)
ls -> libc.so.1:*malloc(0x9000, 0x0, 0x0)
ls -> libc.so.1:*lstat64(0x23ee8, 0xef6ff7a0, 0x0)
ls -> libc.so.1:*printf(0x13a64, 0x26208, 0x23e0)
ls -> libc.so.1:*printf(0x13a64, 0x26448, 0x23e0)
ls -> libc.so.1:*exit(0x0, 0x24220, 0x2421c)

ATTRIBUTES
See attributes(5) for descriptions of the following attributes:
sotruss(1)

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWtoo</td>
</tr>
</tbody>
</table>

SEE ALSO

d.so.1(1), truss(1), whocalls(1), fork(2), attributes(5)

*Linker and Libraries Guide*
The spell command collects words from the named files and looks them up in a spelling list. Words that neither occur among nor are derivable (by applying certain inflections, prefixes, or suffixes) from words in the spelling list are written to the standard output.

If there are no file arguments, words to check are collected from the standard input. spell ignores most troff(1), tbl(1), and eqn(1) constructs. Copies of all output words are accumulated in the history file (spellhist), and a stop list filters out misspellings (for example, their=thy-y-ier) that would otherwise pass.

By default, spell (like deroff(1)) follows chains of included files (.so and .nx troff(1) requests), unless the names of such included files begin with /usr/lib.

The standard spelling list is based on many sources, and while more haphazard than an ordinary dictionary, is also more effective in respect to proper names and popular technical words. Coverage of the specialized vocabularies of biology, medicine and chemistry is light.

Three programs help maintain and check the hash lists used by spell:

hashmake  
Reads a list of words from the standard input and writes the corresponding nine-digit hash code on the standard output.

spellin  
Reads n hash codes from the standard input and writes a compressed spelling list on the standard output.

hashcheck  
Reads a compressed spelling_list and recreates the nine-digit hash codes for all the words in it. It writes these codes on the standard output.

The following options are supported:

-b  
Check British spelling. Besides preferring "centre," "colour," "programme," "speciality," "travelled," and so forth, this option insists upon -ise in words like "standardise."

-i  
Cause deroff(1) to ignore .so and .nx commands. If deroff(1) is not present on the system, then this option is ignored.

-l  
Follow the chains of all included files.

-v  
Print all words not literally in the spelling list, as well as plausible derivations from the words in the spelling list.
Print every plausible stem, one per line, with = preceding each word.

Specify a set of words that are correct spellings (in addition to spell’s own spelling list) for each job. local_file is the name of a user-provided file that contains a sorted list of words, one per line. Words found in local_file are removed from spell’s output. Use sort(1) to order local_file in ASCII collating sequence. If this ordering is not followed, some entries in local_file may be ignored.

The following operands are supported:

file A path name of a text file to check for spelling errors. If no files are named, words are collected from the standard input.

See environ(5) for descriptions of the following environment variables that affect the execution of spell: LC_CTYPE, LC_MESSAGES, and NLSPATH.

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

D_SPELL=/usr/lib/spell/hlist[ab] hashed spelling lists, American & British

S_SPELL=/usr/lib/spell/hstop hashed stop list

H_SPELL=/var/adm/spellhist history file

/usr/share/lib/dict/words master dictionary

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
</tbody>
</table>

Misspelled words can be monitored by default by setting the H_SPELL variable in /usr/bin/spell to the name of a file that has permission mode 666.

spell works only on English words defined in the U.S. ASCII codeset.

Because copies of all output are accumulated in the spellhist file, spellhist may grow quite large and require purging.
The spelling list’s coverage is uneven; new installations may wish to monitor the output for several months to gather local additions.

British spelling was done by an American.
NAME | spline – interpolate smooth curve

SYNOPSIS | **spline** [-aknpx] ...

DESCRIPTION | **spline** takes pairs of numbers from the standard input as abcissas and ordinates of a function. It produces a similar set, which is approximately equally spaced and includes the input set, on the standard output. The cubic spline output (R. W. Hamming, *Numerical Methods for Scientists and Engineers*, 2nd ed., 349ff) has two continuous derivatives, and sufficiently many points to look smooth when plotted, for example by **graph**(1).

OPTIONS | -a  Supply abscissas automatically (they are missing from the input); spacing is given by the next argument, or is assumed to be 1 if next argument is not a number.

          -k  The constant $k$ used in the boundary value computation

          \[ (2\text{nd deriv. at end}) = k(2\text{nd deriv. next to end}) \]

          is set by the next argument. By default $k = 0$.

          -n  Space output points so that approximately $n$ intervals occur between the lower and upper $x$ limits. (Default $n = 100$.)

          -p  Make output periodic, that is, match derivatives at ends. First and last input values should normally agree.

          -x  Next 1 (or 2) arguments are lower (and upper) $x$ limits. Normally these limits are calculated from the data. Automatic abscissas start at lower limit (default 0).

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
</tbody>
</table>

SEE ALSO | **graph**(1), **attributes**(5)


DIAGNOSTICS | When data is not strictly monotonic in $x$, **spline** reproduces the input without interpolating extra points.

BUGS | A limit of 1000 input points is enforced silently.
split(1)

NAME  split — split a file into pieces

SYNOPSIS  split  [-linecount | -l linecount] [-a suffixlength] [file [name]]
            split  [-b n | nk | nm] [-a suffixlength] [file [name]]

DESCRIPTION  The split utility reads file and writes it in linecount-line pieces into a set of output-files. The name of the first output-file is name with aa appended, and so on lexicographically, up to zz (a maximum of 676 files). The maximum length of name is 2 characters less than the maximum filename length allowed by the filesystem. See statvfs(2). If no output name is given, x is used as the default (output-files will be called xaa, xab, and so forth).

OPTIONS  The following options are supported:

  -linecount | -l linecount  Number of lines in each piece. Defaults to 1000 lines.

  -a suffixlength  Uses suffixlength letters to form the suffix portion of the filenames of the split file. If -a is not specified, the default suffix length is 2. If the sum of the name operand and the suffixlength option-argument would create a filename exceeding NAME_MAX bytes, an error will result; split will exit with a diagnostic message and no files will be created.

  -b n  Splits a file into pieces n bytes in size.

  -b nk  Splits a file into pieces n*1024 bytes in size.

  -b nm  Splits a file into pieces n*1 048 576 bytes in size.

OPERANDS  The following operands are supported:

  file  The path name of the ordinary file to be split. If no input file is given or file is -, the standard input will be used.

  name  The prefix to be used for each of the files resulting from the split operation. If no name argument is given, x will be used as the prefix of the output files. The combined length of the basename of prefix and suffixlength cannot exceed NAME_MAX bytes; see OPTIONS.

USAGE  See largefile(5) for the description of the behavior of split when encountering files greater than or equal to 2 Gbyte (2^31 bytes).

ENVIRONMENT VARIABLES  See environ(5) for descriptions of the following environment variables that affect the execution of split: LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS  The following exit values are returned:

  0  Successful completion.

  >0  An error occurred.

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

1424  man pages section 1: User Commands • Last Revised 16 Apr 1999
split(1)

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

SEE ALSO  csplit(1), statvfs(2), attributes(5), environ(5), largefile(5)
srchtxt(1)

NAME srchtxt – display contents of, or search for a text string in, message data bases

SYNOPSIS srchtxt [-s] [-l locale] [-m msgfile, ...] [text]

DESCRIPTION The srchtxt utility is used to display all the text strings in message data bases, or to search for a text string in message data bases (see mkmsgs(1)). These data bases are files in the directory /usr/lib/locale/locale/LC_MESSAGES (see setlocale(3C)), unless a file name given with the -m option contains a /. The directory locale can be viewed as the name of the language in which the text strings are written. If the -l option is not specified, the files accessed will be determined by the value of the environment variable LC_MESSAGES. If LC_MESSAGES is not set, the files accessed will be determined by the value of the environment variable LANG. If LANG is not set, the files accessed will be in the directory /usr/lib/locale//C/LC_MESSAGES, which contains default strings.

If no text argument is present, then all the text strings in the files accessed will be displayed.

If the -s option is not specified, the displayed text is prefixed by message sequence numbers. The message sequence numbers are enclosed in angle brackets: <msgfile:msgnum>.

msgfile name of the file where the displayed text occurred
msgnum sequence number in msgfile where the displayed text occurred

This display is in the format used by gettext(1) and gettext(3C).

OPTIONS

-s Suppress printing of the message sequence numbers of the messages being displayed.

-l locale Access files in the directory /usr/lib/locale/locale/LC_MESSAGES. If -m msgfile is also supplied, locale is ignored for msgfiles containing a /.

-m msgfile Access files specified by one or more msgfiles. If msgfile contains a / character, then msgfile is interpreted as a pathname; otherwise, it will be assumed to be in the directory determined as described above. To specify more than one msgfile, separate the file names using commas.

text Search for the text string specified by text and display each one that matches. text can take the form of a regular expression; see regexp(5).

EXAMPLES

EXAMPLE 1 Using srchtxt

If message files have been installed in a locale named french by using mkmsgs(1), then you could display the entire set of text strings in the french locale (/usr/lib/locale/french/LC_MESSAGES/*) by typing:

eexample% srchtxt -l french
EXAMPLE 1 Using `srchtxt`  

EXAMPLE 2 Using `srchtxt`  

If a set of error messages associated with the operating system have been installed in the file `UX` in the french locale (`/usr/lib/locale/french/LC_MESSAGE/UX`), then, using the value of the `LANG` environment variable to determine the locale to be searched, you could search that file in that locale for all error messages dealing with files by typing:

```bash
example$ setenv LANG=french; export LANG
example$ srchtxt -m UX "[Ff]ichier"
```

If `/usr/lib/locale/french/LC_MESSAGES/UX` contained the following strings:

```
Erreur E/S
Liste d'arguments trop longue
Fichier inexistant
Argument invalide
Trop de fichiers ouverts
Fichier trop long
Trop de liens
Argument hors du domaine
Identificateur supprim
Etreinte fatale
```

then the following strings would be displayed:

```
<UX:3>Fichier inexistant
<UX:5>Trop de fichiers ouverts
<UX:6>Fichier trop long
```

EXAMPLE 3 Using `srchtxt`  

If a set of error messages associated with the operating system have been installed in the file `UX` and a set of error messages associated with the INGRESS data base product have been installed in the file `ingress`, both in the german locale, then you could search for the pattern `[Dd]atei` in both the files `UX` and `ingress` in the german locale by typing:

```bash
example$ srchtxt -l german -m UX,ingress "[Dd]atei"
```

See `environ(5)` for a description of the `LC_CTYPE` environment variable that affects the execution of `srchtxt`.

FILES  

```
/usr/lib/locale/C/LC_MESSAGES/*  
default files created by `mkmsgs(1)`
/usr/lib/locale/locale/LC_MESSAGES/*  
message files created by `mkmsgs(1)`
```
srchtxt(1)

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWloc</td>
</tr>
</tbody>
</table>

SEE ALSO

exstr(1), gettext(1), locale(1), msgs(1), gettext(3C), setlocale(3C), attributes(5), environ(5), locale(5), regexp(5)

DIAGNOSTICS

The error messages produced by srchtxt are intended to be self-explanatory. They indicate an error in the command line or errors encountered while searching for a particular locale and/or message file.
**NAME**

ssh – OpenSSH secure shell client (remote login program)

**SYNOPSIS**

```
ssh [-l login_name] [ hostname | user@hostname ] [ command ]
```

```
ssh -afgknqtvxACNPTX246 [-c cipher_spec] [-e escape_char] [-i identity_file]
[ -l login_name ] [ -o option ] [ -p port ] [ -L port:host:hostport ] [ -R port:host:hostport ] [ hostname | user@hostname ] [ command ]
```

**DESCRIPTION**

ssh (Secure Shell) is a program for logging into a remote machine and for executing commands on a remote machine. It is intended to replace rlogin and rsh, and to provide secure encrypted communications between two untrusted hosts over an insecure network. X11 connections and arbitrary TCP/IP ports can also be forwarded over the secure channel.

ssh connects and logs into the specified hostname. The user must prove his or her identity to the remote machine using one of several methods depending on the protocol version used:

**SSH protocol version 1**

First, if the machine the user logs in from is listed in `/etc/hosts.equiv` or `/etc/shosts.equiv` on the remote machine, and the user names are the same on both sides, the user is immediately permitted to log in. Second, if `.rhosts` or `.shosts` exists in the user’s home directory on the remote machine and contains a line containing the name of the client machine and the name of the user on that machine, the user is permitted to log in. This form of authentication alone is normally not allowed by the server because it is not secure.

The second (and primary) authentication method is the rhosts or hosts.equiv method combined with RSA-based host authentication. It means that if the login would be permitted by `$HOME/.rhosts`, `$HOME/.shosts`, `/etc/hosts.equiv`, or `/etc/shosts.equiv`, and if additionally the server can verify the client’s host key (see `/etc/ssh_known_hosts` in the FILES section), only then is login permitted. This authentication method closes security holes due to IP spoofing, DNS spoofing, and routing spoofing.

Note to the administrator: `/etc/hosts.equiv`, `$HOME/.rhosts`, and the rlogin/rsh protocol in general, are inherently insecure and should be disabled if security is desired.

As a third authentication method, ssh supports RSA-based authentication. The scheme is based on public-key cryptography. There are cryptosystems where encryption and decryption are done using separate keys, and it is not possible to derive the decryption key from the encryption key. RSA is one such system. The idea is that each user creates a public/private key pair for authentication purposes. The server knows the public key, and only the user knows the private key. The file `$HOME/.ssh/authorized_keys` lists the public keys that are permitted for logging in. When the user logs in, the ssh program tells the server which key pair it would like to use for authentication. The server checks if this key is permitted, and if so, sends the user (actually the ssh program running on behalf of the user) a challenge in
the form of a random number, encrypted by the user’s public key. The challenge can only be decrypted using the proper private key. The user’s client then decrypts the challenge using the private key, proving that he or she knows the private key but without disclosing it to the server.

**ssh** implements the RSA authentication protocol automatically. The user creates his or her RSA key pair by running `ssh-keygen(1)`. This stores the private key in `$HOME/.ssh/identity` and the public key in `$HOME/.ssh/identity.pub` in the user’s home directory. The user should then copy the `identity.pub` to `$HOME/.ssh/authorized_keys` in his or her home directory on the remote machine (the `authorized_keys` file corresponds to the conventional `$HOME/.rhosts` file, and has one key per line, though the lines can be very long). After this, the user can log in without giving the password. RSA authentication is much more secure than `rhosts` authentication.

The most convenient way to use RSA authentication may be with an authentication agent. See `ssh-agent(1)` for more information.

If other authentication methods fail, `ssh` prompts the user for a password. The password is sent to the remote host for checking. However, since all communications are encrypted, the password cannot be seen by someone listening on the network.

When a user connects using the protocol version 2, different authentication methods are available. At first, the client attempts to authenticate using the public key method. If this method fails, password authentication is tried.

The public key method is similar to RSA authentication described in the previous section except that the DSA algorithm is used instead of the patented RSA algorithm. The client uses his private DSA key `$HOME/.ssh/id_dsa` to sign the session identifier and sends the result to the server. The server checks whether the matching public key is listed in `$HOME/.ssh/authorized_keys` and grants access if both the key is found and the signature is correct. The session identifier is derived from a shared Diffie-Hellman value and is known only to the client and the server.

If public key authentication fails or is not available, a password can be sent encrypted to the remote host for proving the user’s identity. This protocol 2 implementation does not yet support Kerberos or S/Key authentication.

Protocol 2 provides additional mechanisms for confidentiality (the traffic is encrypted using 3DES, Blowfish, CAST128 or Arcfour) and integrity (`hmac-sha1`, `hmac-md5`). Notice that protocol 1 lacks a strong mechanism for ensuring the integrity of the connection.

When the user’s identity has been accepted by the server, the server either executes the given command, or logs into the machine and gives the user a normal shell on the remote machine. All communication with the remote command or shell will be automatically encrypted.
If a pseudo-terminal has been allocated (normal login session), the user can disconnect with `~`, and suspend `ssh` with `~^Z`. All forwarded connections can be listed with `~#`. If the session blocks waiting for forwarded X11 or TCP/IP connections to terminate, `ssh` can be backgrounded with `~&`, although this should not be used while the user shell is active, as it can cause the shell to hang. All available escapes can be listed with `~?`.

A single tilde character can be sent as `~~` (or by following the tilde by a character other than those described above). The escape character must always follow a newline to be interpreted as special. The escape character can be changed in configuration files or on the command line.

If no pseudo tty has been allocated, the session is transparent and can be used to reliably transfer binary data. On most systems, setting the escape character to "none" will also make the session transparent even if a tty is used.

The session terminates when the command or shell in the remote machine exits and all X11 and TCP/IP connections have been closed. The exit status of the remote program is returned as the exit status of `ssh`.

If the user is using X11 (the `DISPLAY` environment variable is set), the connection to the X11 display is automatically forwarded to the remote side in such a way that any X11 programs started from the shell (or command) will go through the encrypted channel, and the connection to the real X server will be made from the local machine. The user should not manually set `DISPLAY`. Forwarding of X11 connections can be configured on the command line or in configuration files.

The `DISPLAY` value set by `ssh` will point to the server machine, but with a display number greater than zero. This is normal behavior, because `ssh` creates a “proxy” X server on the server machine for forwarding the connections over the encrypted channel.

`ssh` will also automatically set up Xauthority data on the server machine. For this purpose, it will generate a random authorization cookie, store it in Xauthority on the server, and verify that any forwarded connections carry this cookie and replace it by the real cookie when the connection is opened. The real authentication cookie is never sent to the server machine (and no cookies are sent in the plain).

If the user is using an authentication agent, the connection to the agent is automatically forwarded to the remote side unless disabled on the command line or in a configuration file.

Forwarding of arbitrary TCP/IP connections over the secure channel can be specified either on the command line or in a configuration file. One possible application of TCP/IP forwarding is a secure connection to an electronic purse. Another possible application is going through firewalls.

`ssh` automatically maintains and checks a database containing identifications for all hosts it has ever been used with. RSA host keys are stored in `$HOME/.ssh/known_hosts` in the user’s home directory. Additionally, the file
/etc/ssh_known_hosts is automatically checked for known hosts. Any new hosts are automatically added to the user's file. If a host's identification ever changes, ssh warns about this and disables password authentication to prevent a trojan horse from getting the user's password. Another purpose of this mechanism is to prevent man-in-the-middle attacks which could otherwise be used to circumvent the encryption. The StrictHostKeyChecking option (see below) can be used to prevent logins to machines whose host key is not known or has changed.

**OPTIONS**

The following options are supported:

- `-2`
  Forces ssh to try protocol version 2 only.

- `-4`
  Forces ssh to use IPv4 addresses only.

- `-6`
  Forces ssh to use IPv6 addresses only.

- `-a`
  Disables forwarding of the authentication agent connection.

- `-A`
  Enables forwarding of the authentication agent connection. This can also be specified on a per-host basis in a configuration file.

- `-c blowfish | 3des`
  Selects the cipher to use for encrypting the session. 3des is used by default. It is believed to be secure. 3des (triple-des) is an encrypt-decrypt-encrypt triple with three different keys. It is presumably more secure than the des cipher, which is no longer fully supported in ssh. blowfish is a fast block cipher, it appears very secure and is much faster than 3des.

- `-c 3des-cbc,blowfish-cbc,aes128-cbc`
  Additionally, for protocol version 2 a comma-separated list of ciphers can be specified in order of preference. Protocol version 2 supports 3DES, Blowfish, and AES 128 in CBC mode.

- `-C`
  Requests compression of all data (including stdin, stdout, stderr, and data for forwarded X11 and TCP/IP connections). The compression algorithm is the same used by gzip(1). (The gzip man page is available in the SUNWsfman package.) The "level" can be controlled by the CompressionLevel option (see below). Compression is desirable on modem lines and other slow connections, but will only slow down things on fast networks. The default value can be set on a host-by-host basis in the configuration files. See the Compress option below.

- `-e ch | ^ch | none`
  Sets the escape character for sessions with a pty (default: ‘~’). The escape character is only recognized at the beginning of a line. The escape character followed by a dot (‘.’) closes the connection. If followed by control-Z, the escape character suspends
the connection. If followed by itself, the escape character sends itself once. Setting
the character to "none" disables any escapes and makes the session fully
transparent.

- `f`
  Requests `ssh` to go to background just before command execution. This is useful if
  `ssh` is going to ask for passwords or passphrases, but the user wants it in the
  background. This implies the `-n` option. The recommended way to start X11
  programs at a remote site is with something like `ssh -f host xterm`.

- `g`
  Allows remote hosts to connect to local forwarded ports.

- `i identity_file`
  Selects the file from which the identity (private key) for RSA authentication is read.
  Default is `$HOME/.ssh/identity` in the user's home directory. Identity files may
  also be specified on a per-host basis in the configuration file. It is possible to have
  multiple `-i` options (and multiple identities specified in configuration files).

- `l login_name`
  Specifies the user to log in as on the remote machine. This also may be specified on
  a per-host basis in the configuration file.

- `L port:host:port`
  Specifies that the given port on the local (client) host is to be forwarded to the given
  host and port on the remote side. This works by allocating a socket to listen to the
  port on the local side. Then, whenever a connection is made to this port, the
  connection is forwarded over the secure channel and a connection is made to `host
  port hostport` from the remote machine. Port forwardings can also be specified in the
  configuration file. Only root can forward privileged ports. IPv6 addresses can be
  specified with an alternative syntax: `port/host/hostport`.

- `n`
  Redirects stdin from `/dev/null` (actually, prevents reading from stdin). This must
  be used when `ssh` is run in the background. A common trick is to use this to run
  X11 programs on a remote machine. For example,

  `ssh -n shadows.cs.hut.fi emacs &`

  will start an emacs on `shadows.cs.hut.fi`, and the X11 connection will be
  automatically forwarded over an encrypted channel. The `ssh` program will be put
  in the background. This does not work if `ssh` needs to ask for a password or
  passphrase. See also the `-f` option.)

- `N`
  Does not execute a remote command. This is useful if you just want to forward
  ports (protocol version 2 only).

- `o option`
  Can be used to give options in the format used in the configuration file. This is
  useful for specifying options for which there is no separate command-line flag. The
  option has the same format as a line in the configuration file.
**ssh(1)**

- **-p port**  
  Specifies the port to connect to on the remote host. This can be specified on a per-host basis in the configuration file.

- **-P**  
  Uses a non-privileged port for outgoing connections. This can be used if your firewall does not permit connections from privileged ports. Notice that this option turns off `RhostsAuthentication` and `RhostsRSAAuthentication`.

- **-q**  
  Quiet mode. Causes all warning and diagnostic messages to be suppressed. Only fatal errors are displayed.

- **-R port:host:hostport**  
  Specifies that the given port on the remote (server) host is to be forwarded to the given host and port on the local side. This works by allocating a socket to listen to the port on the remote side. Then, whenever a connection is made to this port, the connection is forwarded over the secure channel and a connection is made to `host port hostport` from the local machine. Port forwardings can also be specified in the configuration file. Privileged ports can be forwarded only when logging in as root on the remote machine.

- **-t**  
  Forces pseudo-tty allocation. This can be used to execute arbitrary screen-based programs on a remote machine, which can be very useful, for example, when implementing menu services.

- **-T**  
  Disables pseudo-tty allocation (protocol version 2 only).

- **-v**  
  Verbose mode. Causes `ssh` to print debugging messages about its progress. This is helpful in debugging connection, authentication, and configuration problems. Multiple `-v` options increase the verbosity. Maximum is 3.

- **-x**  
  Disables X11 forwarding.

- **-X**  
  Enables X11 forwarding. This can also be specified on a per-host basis in a configuration file.

**ENVIRONMENT VARIABLES**

`ssh` will normally set the following environment variables:

- **DISPLAY**  
  The `DISPLAY` variable indicates the location of the X11 server. It is automatically set by `ssh` to point to a value of the form `hostname:n` where `hostname` indicates the host where the shell runs, and `n` is an integer greater than or equal to 1. `ssh` uses this special value to forward X11 connections over the secure channel. The user should normally not set `DISPLAY` explicitly, as that will render the X11 connection insecure (and will require the user to manually copy any required authorization cookies).
HOME  Set to the path of the user’s home directory.
LOGNAME  Synonym for USER. Set for compatibility with systems that use this variable.
MAIL  Set to point to the user’s mailbox.
PATH  Set to the default PATH, as specified when compiling ssh.
SSH_AUTH_SOCK  Indicates the path of a unix-domain socket used to communicate with the agent.
SSH_CLIENT  Identifies the client end of the connection. The variable contains three space-separated values: client ip-address, client port number, and server port number.
SSH_TTY  This is set to the name of the tty (path to the device) associated with the current shell or command. If the current session has no tty, this variable is not set.
TZ  The timezone variable is set to indicate the present timezone if it was set when the daemon was started, that is, the daemon passes the value on to new connections.
USER  Set to the name of the user logging in.

Additionally, ssh reads $HOME/.ssh/environment and adds lines of the format VARNAME=value to the environment.

EXIT STATUS
The following exit values are returned:
0  Successful completion.
1  An error occurred.

FILES
$HOME/.ssh/known_hosts
Records host keys for all hosts the user has logged into that are not in /etc/ssh_known_hosts. See sshd(1M).

$HOME/.ssh/identity
$HOME/.ssh/id_dsa
Contains the RSA and the DSA authentication identity of the user. These files contain sensitive data and should be readable by the user but not accessible by others (read/write/execute). Notice that ssh ignores a private key file if it is accessible by others. It is possible to specify a passphrase when generating the key. The passphrase will be used to encrypt the sensitive part of this file using 3DES.

$HOME/.ssh/identity.pub
$HOME/.ssh/id_dsa.pub
Contains the public key for authentication, that is, the public part of the identity file in human-readable form. The contents of the $HOME/.ssh/identity.pub file should be added to $HOME/.ssh/authorized_keys on all machines where you wish to log in using RSA authentication. The contents of the $HOME/.ssh/id_dsa.pub file should be added to
$HOME/.ssh/authorized_keys on all machines where you wish to log in using DSA authentication. These files are not sensitive and can, but need not, be readable by anyone. These files are never used automatically and are not necessary. They are provided only for the convenience of the user.

$HOME/.ssh/config

This is the per-user configuration file. The format of this file is described above. This file is used by the ssh client. This file does not usually contain any sensitive information, but the recommended permissions are read/write for the user and not accessible by others.

$HOME/.ssh/authorized_keys

Lists the DSA keys that can be used for logging in as this user. This file is not highly sensitive, but the recommended permissions are read/write for the user and not accessible by others.

/etc/ssh/ssh_known_hosts

Systemwide list of known host keys. /etc/ssh_known_hosts contains RSA keys. This file should be prepared by the system administrator to contain the public host keys of all machines in the organization and should be world-readable. The file contains public keys, one per line, in the following format, with fields separated by spaces: system name, number of bits in modulus, public exponent, modulus, and optional comment field. When different names are used for the same machine, all such names should be listed, separated by commas. See sshd(1M).

The canonical system name (as returned by name servers) is used by sshd(1M) to verify the client host when logging in. Other names are needed because ssh does not convert the user-supplied name to a canonical name before checking the key, to prevent someone with access to the name servers from being able to fool host authentication.

/etc/ssh/ssh_config

Systemwide configuration file. This file provides defaults for those values that are not specified in the user’s configuration file, and for those users who do not have a configuration file.

This file must be world-readable.

$HOME/.rhosts

This file is used in .rhosts authentication to list the host/user pairs that are permitted to log in. (Notice that this file is also used by rlogin and rsh, which makes using this file insecure.) Each line of the file contains a host name (in the canonical form returned by name servers), and then a user name on that host, separated by a space. On some machines, this file may need to be world-readable if the user’s home directory is on an NFS partition, because sshd(1M) reads it as root. Additionally, this file must be owned by the user and must not have write permissions for anyone else. The recommended permission for most machines is read/write for the user and not accessible by others.
Notice that, by default, `sshd(1M)` will be installed so that it requires successful RSA host authentication before permitting `.rhosts` authentication. If your server machine does not have the client’s host key in `/etc/ssh_known_hosts`, you can store it in `$HOME/.ssh/known_hosts`. The easiest way to do this is to connect back to the client from the server machine using `ssh`. This will automatically add the host key to `$HOME/.ssh/known_hosts`.

`$HOME/.ssh/hosts`

This file is used exactly the same way as `.rhosts`. The purpose for having this file is to be able to use rhosts authentication with `ssh` without permitting login with `rlogin(1)` or `rsh(1)`.

`/etc/hosts.equiv`

This file is used during `.rhosts` authentication. It contains canonical hosts names, one per line. (See `sshd(1M)` for the full format description.). If the client host is found in this file, login is automatically permitted, provided that client and server user names are the same. In addition, successful RSA host authentication is normally required. This file should only be writable by root.

`/etc/ssh/hosts.equiv`

This file is processed exactly as `/etc/hosts.equiv`. This file may be useful to permit logins using `ssh` but not using `rsh` or `rlogin`.

`/etc/ssh/sshrc`

Commands in this file are executed by `ssh` when the user logs in just before the user’s shell or command is started. See `sshd(1M)` for more information.

`$HOME/.ssh/rc`

Commands in this file are executed by `ssh` when the user logs in just before the user’s shell or command is started. See `sshd(1M)` for more information.

`$HOME/.ssh/environment`

Contains additional definitions for environment variables. See `ENVIRONMENT VARIABLES`.

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsshu</td>
</tr>
</tbody>
</table>

**SEE ALSO**

`rlogin(1), rsh(1), scp(1), ssh-add(1), ssh-agent(1), ssh-keygen(1), telnet(1), sshd(1M), ssh_config(4), attributes(5)`

To view license terms, attribution, and copyright for OpenSSH, the default path is `/var/sadm/pkg/SUNWsshdr/install/copyright`. If the Solaris operating environment has been installed anywhere other than the default, modify the given path to access the file at the installed location.
ssh(1)

AUTHORS
OpenSSH is a derivative of the original and free ssh 1.2.12 release by Tatu Ylonen. Aaron Campbell, Bob Beck, Markus Friedl, Niels Provos, Theo de Raadt and Dug Song removed many bugs, added newer features and created Open SSH. Markus Friedl contributed the support for SSH protocol versions 1.4 and 2.0.
**NAME**  
ssh-add – add RSA or DSA identities for the authentication agent

**SYNOPSIS**  
```bash
ssh-add [-ldD] [ file ]
```

**DESCRIPTION**  
The `ssh-add` utility adds RSA or DSA identities to the authentication agent, `ssh-agent`(1). When run without arguments, it attempts to add all of the files `$HOME/.ssh/identity` (RSA v1), `$HOME/.ssh/id_rsa` (RSA v2), and `$HOME/.ssh/id_dsa` (DSA v2) that exist. If more than one of the private keys exists, an attempt to decrypt each with the same passphrase will be made before reprompting for a different passphrase. The passphrase is read from the user’s tty or by running the program defined in `SSH_ASKPASS` (see below).

The authentication agent must be running.

**OPTIONS**  
The following options are supported:

- `-d` Instead of adding the identity, this option **removes** the identity from the agent.
- `-D` Deletes all identities from the agent.
- `-l` Lists fingerprints of all identities currently represented by the agent.
- `-L` Lists public key parameters of all identities currently represented by the agent.

**ENVIRONMENT VARIABLES**  
`DISPLAY`  
If `ssh-add` needs a passphrase, it will read the passphrase from the current terminal if it was run from a terminal. If `ssh-add` does not have a terminal associated with it but `DISPLAY` and `SSH_ASKPASS` are set, it will execute the program specified by `SSH_ASKPASS` and open an X11 window to read the passphrase. This is particularly useful when calling `ssh-add` from a `.Xsession` or related script.

`SSH_ASKPASS`  
These files should not be readable by anyone but the user. Notice that `ssh-add` ignores a file if it is accessible by others. It is possible to specify a passphrase when generating the key; that passphrase will be used to encrypt the private part of this file.

If these files are stored on a network file system it is assumed that either the protection provided in the file themselves or the transport layer of the network file system provides sufficient protection for the site policy. If this is not the case, then it is recommended the key files are stored on removable media or locally on the relevant hosts.

Recommended names for the DSA and RSA key files:

**FILES**

User Commands  1439
ssh-add(1)

$HOME/.ssh/identity Contains the RSA authentication identity of the user for protocol version 1.

$HOME/.ssh/identity.pub Contains the public part of the RSA authentication identity of the user for protocol version 1.

$HOME/.ssh/id_dsa Contains the private DSA authentication identity of the user.

$HOME/.ssh/id_dsa.pub Contains the public part of the DSA authentication identity of the user.

$HOME/.ssh/id_rsa Contains the private RSA authentication identity of the user.

$HOME/.ssh/id_rsa.pub Contains the public part of the RSA authentication identity of the user.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsshu</td>
</tr>
</tbody>
</table>

SEE ALSO ssh(1), ssh-agent(1), ssh-keygen(1), sshd(1M), attributes(5)

To view license terms, attribution, and copyright for OpenSSH, the default path is /var/sadm/pkg/SUNWsshdr/install/copyright. If the Solaris operating environment has been installed anywhere other than the default, modify the given path to access the file at the installed location.

AUTHORS OpenSSH is a derivative of the original and free ssh 1.2.12 release by Tatu Ylonen. Aaron Campbell, Bob Beck, Markus Friedl, Niels Provos, Theo de Raadt and Dug Song removed many bugs, added newer features and created Open SSH. Markus Friedl contributed the support for SSH protocol versions 1.4 and 2.0.
ssh-agent is a program to hold private keys used for public key authentication (RSA, DSA). ssh-agent is often started at the beginning of a login session. All other windows or programs are started as clients to the ssh-agent program. Through use of environment variables, the agent can be located and automatically used for authentication when logging in to other machines using ssh(1). (See System Administration Guide: Security Services.)

If a command line is given, this is executed as a subprocess of the agent. When the command dies, so does the agent.

The agent initially does not have any private keys. Keys are added using ssh-add(1), which sends the identity to the agent. Several identities can be stored in the agent; the agent can automatically use any of these identities. Use the -l option in ssh-add(1) to display the identities currently held by the agent.

The agent is run in the user’s local host. Authentication data need not be stored on any other machine, and authentication passphrases never go over the network. However, if the connection to the agent is forwarded over SSH remote logins, the user can use the privileges given by the identities anywhere in the network in a secure way.

There are two main ways to get an agent setup. Either you let the agent start a new subcommand into which some environment variables are exported, or you let the agent print the needed shell commands (either sh(1) or csh(1) syntax can be generated) which can be evalled in the calling shell. Later, use ssh(1) to look at these variables and use them to establish a connection to the agent.

A unix-domain socket is created (/tmp/ssh-XXXXXXX/agent.pid) and the name of this socket is stored in the SSH_AUTH_SOCK environment variable. The socket is made accessible only to the current user. This method is easily abused by root or another instance of the same user.

The SSH_AGENT_PID environment variable holds the agent’s PID.

The agent exits automatically when the command given on the command line terminates.

The following options are supported:

- `c` Generates C-shell commands on stdout. This is the default if SHELL indicates that it is a csh style of shell.
- `k` Kills the current agent (given by the SSH_AGENT_PID environment variable).
- `s` Generates Bourne shell commands on stdout. This is the default if SHELL does not indicate that it is a csh style of shell.

The following exit values are returned:
ssh-agent(1)

0 Successful completion.
1 An error occurred.

FILES

/tmp/ssh-XXXXXXXX/agent.pid
Unix-domain sockets used to contain the connection to the authentication agent.
These sockets should only be readable by the owner. The sockets are removed when
the agent exits.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsshu</td>
</tr>
</tbody>
</table>

SEE ALSO

ssh(1), ssh-add(1), ssh-keygen(1), sshd(1M), attributes(5)

System Administration Guide: Security Services

To view license terms, attribution, and copyright for OpenSSH, the default path is
/var/sadm/pkg/SUNWsshdr/install/copyright. If the Solaris operating
environment has been installed anywhere other than the default, modify the given
path to access the file at the installed location.

AUTHORS

OpenSSH is a derivative of the original and free ssh 1.2.12 release by Tatu Ylonen.
Aaron Campbell, Bob Beck, Markus Friedl, Niels Provos, Theo de Raadt and Dug Song
removed many bugs, added newer features and created Open SSH. Markus Friedl
contributed the support for SSH protocol versions 1.4 and 2.0.
ssh-http-proxy-connect(1)

NAME
ssh-http-proxy-connect – Secure Shell proxy for HTTP

SYNOPSIS

DESCRIPTION
A proxy command for ssh(1) that uses HTTP CONNECT. Typical use is where connections external to a network are only allowed via a proxy web server.

OPTIONS
The following options are supported:

- `h http_proxy_host` Specifies the proxy web server through which to connect. Overrides the HTTPPROXY and http_proxy environment variables if they are set.

- `p http_proxy_port` Specifies the port on which the proxy web server runs. If not specified, port 80 is assumed. Overrides the HTTPPROXYPORT and http_proxy environment variables if they are set.

OPERANDS
The following operands are supported:

- `http_proxy_host` The host name or IP address (IPv4 or IPv6) of the proxy.

- `http_proxy_port` The numeric port number to connect to on `http_proxy_host`.

- `connect_host` The name of the remote host to which the proxy web server is to connect you.

- `connect_port` The numeric port number of the proxy web server to connect you to on `http_proxy_host`.

EXAMPLES
The recommended way to use a proxy connection command is to configure the ProxyCommand in ssh_config(4) (see Example 1 and Example 2). Example 3 shows how the proxy command can be specified on the command line when running ssh(1).

EXAMPLE 1 Setting the proxy from the environment

The following example uses ssh-http-proxy-connect in ssh_config(4) when the proxy is set from the environment:

```
Host playtime.foo.com
   ProxyCommand /usr/lib/ssh/ssh-http-proxy-connect playtime.foo.com 22
```

EXAMPLE 2 Overriding proxy environment variables

The following example uses ssh-http-proxy-connect in ssh_config(4) to override (or if not set) proxy environment variables:

```
Host playtime.foo.com
   ProxyCommand /usr/lib/ssh/ssh-http-proxy-connect -h webcache
```
EXAMPLE 2 Overriding proxy environment variables (Continued)

    -p 8080 playtime.foo.com 22

EXAMPLE 3 Using the command line

The following example uses ssh-http-proxy-connect from the ssh(1) command line:

    example$ ssh -o 'ProxyCommand="/usr/lib/ssh/ssh-http-proxy-connect \n                      -h webcache -p 8080 playtime.foo.com 22"' playtime.foo.com

ENVIRONMENT VARIABLES

    HTTPPROXY          Takes the http_proxy_host operand to specify the default proxy host. Overrides http_proxy if both are set.

    HTTPPROXYPORT     Takes the http_proxy_port operand to specify the default proxy port. Ignored if HTTPPROXY is not set.

    http_proxy        URL format for specifying proxy host and port.

EXIT STATUS

The following exit values are returned:

    0   Successful completion.
    1   An error occurred.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsshu</td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Stable</td>
</tr>
</tbody>
</table>

SEE ALSO

ssh(1), ssh-socks5-proxy-connect(1), ssh_config(4), attributes(5)
The ssh-keygen utility generates and manages authentication keys for ssh(1). ssh-keygen defaults to generating an RSA key for use by protocol 2.0.

Each user wishing to use SSH with RSA or DSA authentication normally runs this once to create the authentication key in $HOME/.ssh/identity or $HOME/.ssh/id_dsa. The system administrator may also use this to generate host keys.

Ordinarily, this program generates the key and asks for a file in which to store the private key. The public key is stored in a file with the same name but with the “.pub” extension appended. The program also asks for a passphrase. The passphrase may be empty to indicate no passphrase (host keys must have empty passphrases), or it may be a string of arbitrary length. Good passphrases are 10-30 characters long and are not simple sentences or otherwise easy to guess. (English prose has only 1-2 bits of entropy per word, and provides very poor passphrases.) The passphrase can be changed later by using the -p option.

There is no way to recover a lost passphrase. If the passphrase is lost or forgotten, you will have to generate a new key and copy the corresponding public key to other machines.

For RSA, there is also a comment field in the key file that is only for convenience to the user to help identify the key. The comment can tell what the key is for, or whatever is useful. The comment is initialized to “user@host” when the key is created, but can be changed using the -c option.

After a key is generated, instructions below detail where to place the keys to activate them.

The following options are supported:
ssh-keygen(1)

-\(b\) bits  Specifies the number of bits in the key to create. The minimum number is 512 bits. Generally, 1024 bits is considered sufficient. Key sizes above that no longer improve security but make things slower. The default is 1024 bits.

-\(c\) Requests changing the comment in the private and public key files. The program will prompt for the file containing the private keys, for the passphrase if the key has one, and for the new comment.

-\(c\) comment  Provides the new comment.

-\(f\)  Specifies the filename of the key file.

-\(l\)  Shows the fingerprint of the specified private or public key file.

-\(n\) new_passphrase  Provides the new passphrase.

-\(p\)  Requests changing the passphrase of a private key file instead of creating a new private key. The program will prompt for the file containing the private key, for the old passphrase, and will prompt twice for the new passphrase.

-\(P\) passphrase  Provides the (old) passphrase.

-\(q\)  Silences ssh-keygen. Used by /etc/rc when creating a new key.

-\(R\)  If RSA support is functional, immediately exits with code 0. If RSA support is not functional, exits with code 1. This flag will be removed once the RSA patent expires.

-\(t\) type  Specifies the algorithm used for the key, where type is one of rsa, dsa, and rsa1. Type rsa1 is used only for the SSHv1 protocol

-\(x\)  Reads a private OpenSSH DSA format file and prints an SSH2-compatible public key to stdout.

-\(X\)  Reads an unencrypted SSH2-compatible private (or public) key file and prints an OpenSSH compatible private (or public) key to stdout.

-\(y\)  Reads a private OpenSSH DSA format file and prints an OpenSSH DSA public key to stdout.

EXIT STATUS  The following exit values are returned:

0  Successful completion.

1  An error occurred.

FILES  

$HOME/.ssh/identity

Contains the RSA authentication identity of the user. This file should not be readable by anyone but the user. It is possible to specify a passphrase when generating the key; that passphrase will be used to encrypt the private part of this
file using 3DES. This file is not automatically accessed by `ssh-keygen`, but it is
offered as the default file for the private key. `sshd(1M)` will read this file when a
login attempt is made.

`$HOME/.ssh/identity.pub`
Contains the public key for authentication. The contents of this file should be added
to `$HOME/.ssh/authorized_keys` on all machines where you wish to log in
using RSA authentication. There is no need to keep the contents of this file secret.

`$HOME/.ssh/id_dsa`
Contains the DSA authentication identity of the user. This file should not be
readable by anyone but the user. It is possible to specify a passphrase when
generating the key; that passphrase will be used to encrypt the private part of this
file using 3DES. This file is not automatically accessed by `ssh-keygen` but it is
offered as the default file for the private key. `sshd(1M)` will read this file when a
login attempt is made.

`$HOME/.ssh/id_dsa.pub`
Contains the public key for authentication. The contents of this file should be added
to `$HOME/.ssh/authorized_keys` on all machines where you wish to log in
using DSA authentication. There is no need to keep the contents of this file secret.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsshcu</td>
</tr>
</tbody>
</table>

**SEE ALSO**
`ssh(1), ssh-add(1), ssh-agent(1), sshd(1M), attributes(5)`

To view license terms, attribution, and copyright for OpenSSH, the default path is
/var/sadm/pkg/SUNWsshdr/install/copyright. If the Solaris operating
environment has been installed anywhere other than the default, modify the given
path to access the file at the installed location.
Name | ssh-socks5-proxy-connect – Secure Shell proxy for SOCKS5

Synopsis | /usr/lib/ssh/ssh-socks5-proxy-connect [-h socks5_proxy_host] [-p socks5_proxy_port] connect_host connect_port

Description | A proxy command for ssh(1) that uses SOCKS5 (RFC 1928). Typical use is where connections external to a network are only allowed via a socks gateway server.

This proxy command does not provide any of the SOCKS5 authentication mechanisms defined in RFC 1928. Only anonymous connections are possible.

Options | The following options are supported:

- **-h socks5_proxy_host** | Specifies the proxy web server through which to connect. Overrides the SOCKS5_SERVER environment variable.

- **-p socks5_proxy_port** | Specifies the port on which the proxy web server runs. If not specified, port 80 is assumed. Overrides the SOCKS5_PORT environment variable.

Operands | The following operands are supported:

- **socks5_proxy_host** | The host name or IP address (IPv4 or IPv6) of the proxy.

- **socks5_proxy_port** | The numeric port number to connect to on socks5_proxy_host.

- **connect_host** | The name of the remote host to which the socks gateway is to connect you.

- **connect_port** | The numeric port number of the socks gateway to connect you to on connect_host.

Examples | The recommended way to use a proxy connection command is to configure the ProxyCommand in ssh_config(4) (see Example 1 and Example 2). Example 3 shows how the proxy command can be specified on the command line when running ssh(1).

Example 1 Setting the proxy from the environment

The following example uses ssh-socks5-proxy-connect in ssh_config(4) when the proxy is set from the environment:

```
Host playtime.foo.com
  ProxyCommand /usr/lib/ssh/ssh-socks5-proxy-connect \ playtime.foo.com 22
```

Example 2 Overriding proxy environment variables

The following example uses ssh-socks5-proxy-connect in ssh_config(4) to override (or if not set) proxy environment variables:
EXAMPLE 2 Overriding proxy environment variables (Continued)

Host playtime.foo.com
ProxyCommand /usr/lib/ssh/ssh-socks5-proxy-connect -h socks-gw \ 
-p 1080 -n 192.168.100.2 playtime.foo.com 22

EXAMPLE 3 Using the command line

The following example uses ssh-socks5-proxy-connect from the ssh(1) command line:

example$ ssh -o'ProxyCommand=/usr/lib/ssh/ssh-socks5-proxy-connect \ 
-h socks-gw -p 1080 playtime.foo.com 22' playtime.foo.com

ENVIRONMENT VARIABLES

SOCKS5_SERVER Takes socks5_proxy_host operand to specify the default proxy host.

SOCKS5_PORT Takes socks5_proxy_port operand to specify the default proxy port.

EXIT STATUS

The following exit values are returned:

0 Successful completion.
1 An error occurred.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsshu</td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Stable</td>
</tr>
</tbody>
</table>

SEE ALSO ssh(1), ssh-http-proxy-connect(1), ssh_config(4), attributes(5)
strchg(1)

NAME  strchg, strconf – change or query stream configuration

SYNOPSIS  

```
strchg -h module1 [, module2...]  
strchg -p [-a | -u module]  
strchg -f filename  
strconf [-m | -t module]
```

DESCRIPTION  These commands are used to alter or query the configuration of the stream associated with the user’s standard input. The `strchg` command pushes modules on and/or pops modules off the stream. The `strconf` command queries the configuration of the stream. Only the super-user or owner of a STREAMS device may alter the configuration of that stream.

Invoked without any arguments, `strconf` prints a list of all the modules in the stream as well as the topmost driver. The list is printed with one name per line where the first name printed is the topmost module on the stream (if one exists) and the last item printed is the name of the driver.

OPTIONS  The following options apply to `strchg` and, `-h`, `-f`, and `-p` are mutually exclusive.

- `-h module1 [, module2...]` 
  Mnemonic for push, pushes modules onto a stream. It takes as arguments the names of one or more pushable streams modules. These modules are pushed in order; that is, `module1` is pushed first, `module2` is pushed second, etc.

- `-p` 
  Mnemonic for pop, pops modules off the stream. With the `-p` option alone, `strchg` pops the topmost module from the stream.

- `-a module` 
  Pop all the modules above the topmost driver off the stream. This option requires the `-p` option.

- `-u module` 
  All modules above, but not including `module` are popped off the stream. This option requires the `-p` option.

- `-f filename` 
  Specify a `filename` that contains a list of modules representing the desired configuration of the stream. Each module name must appear on a separate line where the first name represents the topmost module and the last name represents the module that should be closest to the driver. `strchg` will determine the current configuration of the stream and pop and push the necessary modules in order to end up with the desired configuration.

The following options apply to `strconf` and, `-m` and `-t` are mutually exclusive.

- `-m module` 
  Determine if the named `module` is present on a stream. If it is, `strconf` prints the message `yes` and returns zero. If not,
strconf prints the message no and returns a non-zero value. The -t and -m options are mutually exclusive.

-t module
Print only the topmost module (if one exists). The -t and -m options are mutually exclusive.

EXAMPLE 1 Using the strchg Command

The following command pushes the module ldterm on the stream associated with the user’s standard input:

```
example$ strchg -h ldterm
```

The following command pops the topmost module from the stream associated with /dev/term/24. The user must be the owner of this device or the super user.

```
example$ strchg -p < /dev/term/24
```

If the file fileconf contains the following:

```
ttcompat
ldterm
ptem
```
then the command

```
example$ strchg -f fileconf
```
will configure the user’s standard input stream so that the module ptem is pushed over the driver, followed by ldterm and ttcompat closest to the stream head.

The strconf command with no arguments lists the modules and topmost driver on the stream; for a stream that has only the module ldterm pushed above the zs driver, it would produce the following output:

```
 ldterm
  zs
```

The following command asks if ldterm is on the stream:

```
example$ strconf -m ldterm
```
and produces the following output while returning an exit status of 0:

```
 yes
```

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO attributes(5), streamio(7)
strchg(1)

**DIAGNOSTICS**

`strchg` returns zero on success. It prints an error message and returns non-zero status for various error conditions, including usage error, bad module name, too many modules to push, failure of an ioctl on the stream, or failure to open *filename* from the `-f` option.

`strconf` returns zero on success (for the `-m` or `-t` option, "success" means the named or topmost module is present). It returns a non-zero status if invoked with the `-m` or `-t` option and the module is not present. It prints an error message and returns non-zero status for various error conditions, including usage error or failure of an ioctl on the stream.

**NOTES**

If the user is neither the owner of the stream nor the super-user, the `strchg` command will fail. If the user does not have read permissions on the stream and is not the super user, the `strconf` command will fail.

If modules are pushed in the wrong order, one could end up with a stream that does not function as expected. For ttys, if the line discipline module is not pushed in the correct place, one could have a terminal that does not respond to any commands.
strings(1)

NAME
strings – find printable strings in an object or binary file

SYNOPSIS
strings [-a | -] [-t format | -o] [-n number | -number] [file...]

DESCRIPTION
The strings utility looks for ASCII strings in a binary file. A string is any sequence of
4 or more printing characters ending with a newline or a null character.

strings is useful for identifying random object files and many other things.

OPTIONS
The following options are supported:

- a | - Look everywhere in the file for strings. If this flag is
  omitted, strings only looks in the initialized data
  space of object files.

- n number | -number Use a number as the minimum string length rather than
  the default, which is 4.

- o Equivalent to -t d option.

- t format Write each string preceded by its byte offset from the
  start of the file. The format is dependent on the single
  character used as the format option-argument:

  d The offset will be written in decimal.

  o The offset will be written in octal.

  x The offset will be written in hexadecimal.

OPERANDS
The following operand is supported:

file A path name of a regular file to be used as input. If no file operand is
specified, the strings utility will read from the standard input.

ENVIRONMENT
See environ(5) for descriptions of the following environment variables that affect the
execution of strings: LC_CTYPE, LC_MESSAGES, and NLSPATH.

VARIABLES

EXIT STATUS
The following exit values are returned:

0 Successful completion.

>0 An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWtoo</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

SEE ALSO
od(1), attributes(5), environ(5)
The algorithm for identifying strings is extremely primitive.

For backwards compatibility, the options -a and – are interchangeable.
### NAME
strip – strip symbol table, debugging and line number information from an object file

### SYNOPSIS
/usr/ccs/bin/strip [-blrVx] file...

### DESCRIPTION
The strip command removes the symbol table, debugging information, and line number information from ELF object files. Once this stripping process has been done, no symbolic debugging access will be available for that file; therefore, this command is normally run only on production modules that have been debugged and tested.

If strip is executed on a common archive file (see ar(3HEAD)) in addition to processing the members, strip will remove the archive symbol table. The archive symbol table must be restored by executing the ar(1) command with the -s option before the archive can be linked by the ld(1) command. strip will produce appropriate warning messages when this situation arises.

strip is used to reduce the file storage overhead taken by the object file.

### OPTIONS
The amount of information stripped from the ELF object file can be controlled by using any of the following options:

- **-b**
  Same effect as the default behavior. This option is obsolete and will be removed in the next release.

- **-l**
  Strip line number information only; do not strip the symbol table or debugging information.

- **-r**
  Same effect as the default behavior. This option is obsolete and will be removed in the next release.

- **-V**
  Print, on standard error, the version number of strip.

- **-x**
  Do not strip the symbol table; debugging and line number information may be stripped.

### OPERANDS
The following operand is supported:

- **file**
  A path name referring to an executable file.

### ENVIRONMENT VARIABLES
See environ(5) for descriptions of the following environment variables that affect the execution of strip: LC_CTYPE, LC_MESSAGES, and NLSPATH.

### EXIT STATUS
The following exit values are returned:

- **0**
  Successful completion.

- **>0**
  An error occurred.

### FILES
/tmp/strp* temporary files

### ATTRIBUTES
See attributes(5) for descriptions of the following attributes:
The symbol table section will not be removed if it is contained within a segment, or the file is either a relocatable or dynamic shared object.

The line number and debugging sections will not be removed if they are contained within a segment, or their associated relocation section is contained within a segment.
NAME
  stty – set the options for a terminal

SYNOPSIS
  /usr/bin/stty [-a] [-g]
  /usr/bin/stty [modes]
  /usr/xpg4/bin/stty [-a | -g]
  /usr/xpg4/bin/stty [modes]

DESCRIPTION
  The stty utility sets certain terminal I/O options for the device that is the current
  standard input. Without arguments, stty reports the settings of certain options.

  In this report, if a character is preceded by a caret (^), then the value of that option is
  the corresponding control character (for example, “^h” is Control-H; in this case, recall
  that Control-H is the same as the “back-space” key). The sequence “^^” means that an
  option has a null value.

  See termio(7I) for detailed information about the modes listed from Control Modes
  through Local Modes. For detailed information about the modes listed under
  Hardware Flow Control Modes and Clock Modes, below, see termiox(7I).

  Operands described in the Combination Modes section are implemented using
  options in the earlier sections. Notice that many combinations of options make no
  sense, but no sanity checking is performed. Hardware flow control and clock modes
  options may not be supported by all hardware interfaces.

OPTIONS
  The following options are supported:

    -a  Writes to standard output all of the option settings for the terminal.
    -g  Reports current settings in a form that can be used as an argument to
        another stty command. Emits termios-type output if the underlying
        driver supports it. Otherwise, it emits termio-type output.

OPERANDS
  The following mode operands are supported:

  Control Modes
    parenb(-parenb)   Enable (disable) parity generation and detection.
    parext(-parext)  Enable (disable) extended parity generation and
                    detection for mark and space parity.
    parodd(-parodd)  Select odd (even) parity, or mark (space) parity if
                    parext is enabled.
    cs5 cs6 cs7 cs8  Select character size (see termio(7I)).
    0                Hang up line immediately.
    hupcl(-hupcl)    Hang up (do not hang up) connection on last close.
    hup(-hup)        Same as hupcl(-hupcl).
    cstopb(-cstopb)  Use two (one) stop bits per character.
    cread(-cread)    Enable (disable) the receiver.
### stty(1)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ctscts (-crlscts)</td>
<td>Enable output hardware flow control. Raise the RTS (Request to Send) modem control line. Suspends output until the CTS (Clear to Send) line is raised.</td>
</tr>
<tr>
<td>ctsxoff (-crlsxoff)</td>
<td>Enable input hardware flow control. Raise the RTS (Request to Send) modem control line to receive data. Suspends input when RTS is low.</td>
</tr>
<tr>
<td>clocal (-clocal)</td>
<td>Assume a line without (with) modem control.</td>
</tr>
<tr>
<td>defeucw</td>
<td>Set the widths of multibyte characters to the values defined in the current locale specified by LC_CTYPE. Internally, width is expressed in terms of bytes per character, and screen or display columns per character.</td>
</tr>
<tr>
<td>110 300 600 1200 1800</td>
<td>Set terminal baud rate to the number given, if possible. (All speeds are not supported by all hardware interfaces.)</td>
</tr>
<tr>
<td>2400 4800 9600 19200</td>
<td></td>
</tr>
<tr>
<td>38400 357600 76800 115200 153600</td>
<td></td>
</tr>
<tr>
<td>230400 307200 460800</td>
<td></td>
</tr>
<tr>
<td>ispeed 0 110 300 600 1200 1800 2400 4800 9600 19200</td>
<td>Set terminal input baud rate to the number given, if possible. (Not all hardware supports split baud rates.) If the input baud rate is set to 0, the input baud rate will be specified by the value of the output baud rate.</td>
</tr>
<tr>
<td>38400 57600 76800 115200</td>
<td></td>
</tr>
<tr>
<td>153600 230400 307200 460800</td>
<td></td>
</tr>
<tr>
<td>ospeed 0 110 300 600 1200 1800 2400 4800 9600 19200</td>
<td>Set terminal output baud rate to the number given, if possible. (Not all hardware supports split baud rates.) If the output baud rate is set to 0, the line will be hung up immediately.</td>
</tr>
<tr>
<td>38400 57600 76800 115200</td>
<td></td>
</tr>
<tr>
<td>153600 230400 307200 460800</td>
<td></td>
</tr>
</tbody>
</table>

**Input Modes**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ignbrk (-ignbrk)</td>
<td>Ignore (do not ignore) break on input.</td>
</tr>
<tr>
<td>brkint (-brkint)</td>
<td>Signal (do not signal) INTR on break.</td>
</tr>
<tr>
<td>ignpar (-ignpar)</td>
<td>Ignore (do not ignore) parity errors.</td>
</tr>
<tr>
<td>parmrk (-parmrk)</td>
<td>Mark (do not mark) parity errors (see termio(7I)).</td>
</tr>
<tr>
<td>inpck (-inpck)</td>
<td>Enable (disable) input parity checking.</td>
</tr>
<tr>
<td>istrrip (-istrip)</td>
<td>Strip (do not strip) input characters to seven bits.</td>
</tr>
</tbody>
</table>
**Output Modes**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inlcr (-inlcr)</td>
<td>Map (do not map) NL to CR on input.</td>
</tr>
<tr>
<td>igncr (-igncr)</td>
<td>Ignore (do not ignore) CR on input.</td>
</tr>
<tr>
<td>icrnl (-icrnl)</td>
<td>Map (do not map) CR to NL on input.</td>
</tr>
<tr>
<td>iuclc (-iuclc)</td>
<td>Map (do not map) upper-case alphabetics to lower case on input.</td>
</tr>
<tr>
<td>ixon (-ixon)</td>
<td>Enable (disable) START/STOP output control. Output is stopped by sending STOP control character and started by sending the START control character.</td>
</tr>
<tr>
<td>ixany (-ixany)</td>
<td>Allow any character (only DC1) to restart output.</td>
</tr>
<tr>
<td>ixoff (-ixoff)</td>
<td>Request that the system send (not send) START/STOP characters when the input queue is nearly empty/full.</td>
</tr>
<tr>
<td>imaxbel (-imaxbel)</td>
<td>Echo (do not echo) BEL when the input line is too long.</td>
</tr>
<tr>
<td>opost (-opost)</td>
<td>Post-process output (do not post-process output; ignore all other output modes).</td>
</tr>
<tr>
<td>olcuc (-olcuc)</td>
<td>Map (do not map) lower-case alphabetics to upper case on output.</td>
</tr>
<tr>
<td>onlcr (-onlcr)</td>
<td>Map (do not map) NL to CR-NL on output.</td>
</tr>
<tr>
<td>ocrnl (-ocrnl)</td>
<td>Map (do not map) CR to NL on output.</td>
</tr>
<tr>
<td>onocr (-onocr)</td>
<td>Do not (do) output CRs at column zero.</td>
</tr>
<tr>
<td>onlret (-onlret)</td>
<td>On the terminal NL performs (does not perform) the CR function.</td>
</tr>
<tr>
<td>ofill (-ofill)</td>
<td>Use fill characters (use timing) for delays.</td>
</tr>
<tr>
<td>ofdel (-ofdel)</td>
<td>Fill characters are DELs (NULs).</td>
</tr>
<tr>
<td>cr0 cr1 cr2 cr3</td>
<td>Select style of delay for carriage returns (see termio(7I)).</td>
</tr>
<tr>
<td>nl0 nl1</td>
<td>Select style of delay for line-feeds (see termio(7I)).</td>
</tr>
<tr>
<td>tab0 tab1 tab2 tab3</td>
<td>Select style of delay for horizontal tabs (see termio(7I)).</td>
</tr>
<tr>
<td>bs0 bs1</td>
<td>Select style of delay for backspaces (see termio(7I)).</td>
</tr>
<tr>
<td>ff0 ff1</td>
<td>Select style of delay for form-feeds (see termio(7I)).</td>
</tr>
<tr>
<td>vt0 vt1</td>
<td>Select style of delay for vertical tabs (see termio(7I)).</td>
</tr>
</tbody>
</table>

**Local Modes**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isig (-isig)</td>
<td>Enable (disable) the checking of characters against the special control characters INTR, QUIT, SWTCH, and SUSP.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>icanon</td>
<td>Enable (disable) canonical input (ERASE and KILL processing). Does not set MIN or TIME.</td>
</tr>
<tr>
<td>xcase</td>
<td>Canonical (unprocessed) upper/lower-case presentation.</td>
</tr>
<tr>
<td>echo</td>
<td>Echo back (do not echo back) every character typed.</td>
</tr>
<tr>
<td>echoe</td>
<td>Echo (do not echo) ERASE character as a backspace-space-backspace string.</td>
</tr>
<tr>
<td>echok</td>
<td>Echo (do not echo) NL after KILL character.</td>
</tr>
<tr>
<td>lfkc</td>
<td>The same as echok (-echok); obsolete.</td>
</tr>
<tr>
<td>echonl</td>
<td>Echo (do not echo) NL.</td>
</tr>
<tr>
<td>nofish</td>
<td>Disable (enable) flush after INTR, QUIT, or SUSP.</td>
</tr>
<tr>
<td>stwrap</td>
<td>Disable (enable) truncation of lines longer than 79 characters on a synchronous line.</td>
</tr>
<tr>
<td>tostop</td>
<td>Send (do not send) SIGTTOU when background processes write to the terminal.</td>
</tr>
<tr>
<td>echoctl</td>
<td>Echo (do not echo) control characters as ^char, delete as ^?.</td>
</tr>
<tr>
<td>echoprt</td>
<td>Echo (do not echo) erase character as character is &quot;erased&quot;.</td>
</tr>
<tr>
<td>echoke</td>
<td>BS-SP-BS erase (do not BS-SP-BS erase) entire line on line kill.</td>
</tr>
<tr>
<td>flusho</td>
<td>Output is (is not) being flushed.</td>
</tr>
<tr>
<td>pendin</td>
<td>Retype (do not retype) pending input at next read or input character.</td>
</tr>
<tr>
<td>iexten</td>
<td>Enable (disable) special control characters not currently controlled by icanon, isig, ixon, or ixoff: VEOLZ, VSWITCH, VREPRINT, VDISCARD, VDSUSP, VWERASE, and VLNEXT.</td>
</tr>
<tr>
<td>stflush</td>
<td>Enable (disable) flush on a synchronous line after every write(2).</td>
</tr>
<tr>
<td>stapl</td>
<td>Use application mode (use line mode) on a synchronous line.</td>
</tr>
<tr>
<td>rtsxoff</td>
<td>Enable (disable) RTS hardware flow control on input.</td>
</tr>
</tbody>
</table>

**Hardware Flow Control Modes**
### User Commands

#### Clock Modes

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ctsxon (-ctsxon)</code></td>
<td>Enable (disable) CTS hardware flow control on output.</td>
</tr>
<tr>
<td><code>dtrxoff (-dtrxoff)</code></td>
<td>Enable (disable) DTR hardware flow control on input.</td>
</tr>
<tr>
<td><code>cdxon (-cdxon)</code></td>
<td>Enable (disable) CD hardware flow control on output.</td>
</tr>
<tr>
<td><code>isxoff (-isxoff)</code></td>
<td>Enable (disable) isochronous hardware flow control on input.</td>
</tr>
<tr>
<td><code>xicibrg</code></td>
<td>Get transmit clock from internal baud rate generator.</td>
</tr>
<tr>
<td><code>xctset</code></td>
<td>Get the transmit clock from transmitter signal element timing (DCE source) lead, CCITT V.24 circuit 114, EIA-232-D pin 15.</td>
</tr>
<tr>
<td><code>xcrset</code></td>
<td>Get transmit clock from receiver signal element timing (DCE source) lead, CCITT V.24 circuit 115, EIA-232-D pin 17.</td>
</tr>
<tr>
<td><code>rcibrg</code></td>
<td>Get receive clock from internal baud rate generator.</td>
</tr>
<tr>
<td><code>rcset</code></td>
<td>Get receive clock from transmitter signal element timing (DCE source) lead, CCITT V.24 circuit 114, EIA-232-D pin 15.</td>
</tr>
<tr>
<td><code>rcrset</code></td>
<td>Get receive clock from receiver signal element timing (DCE source) lead, CCITT V.24 circuit 115, EIA-232-D pin 17.</td>
</tr>
<tr>
<td><code>tsetcoff</code></td>
<td>Transmitter signal element timing clock not provided.</td>
</tr>
<tr>
<td><code>tsetcrbrg</code></td>
<td>Output receive baud rate generator on transmitter signal element timing (DTE source) lead, CCITT V.24 circuit 113, EIA-232-D pin 24.</td>
</tr>
<tr>
<td><code>tsetctbrg</code></td>
<td>Output transmit baud rate generator on transmitter signal element timing (DTE source) lead, CCITT V.24 circuit 113, EIA-232-D pin 24.</td>
</tr>
<tr>
<td><code>tsetctset</code></td>
<td>Output transmitter signal element timing (DCE source) on transmitter signal element timing (DTE source) lead, CCITT V.24 circuit 113, EIA-232-D pin 24.</td>
</tr>
<tr>
<td><code>tsetcrset</code></td>
<td>Output receiver signal element timing (DCE source) on transmitter signal element timing (DTE source) lead, CCITT V.24 circuit 113, EIA-232-D pin 24.</td>
</tr>
<tr>
<td><code>rsetcoff</code></td>
<td>Receiver signal element timing clock not provided.</td>
</tr>
<tr>
<td><code>rsetcrbrg</code></td>
<td>Output receive baud rate generator on receiver signal element timing (DTE source) lead, CCITT V.24 circuit 128, no EIA-232-D pin.</td>
</tr>
<tr>
<td><code>rsetctbrg</code></td>
<td>Output transmit baud rate generator on receiver signal element timing (DTE source) lead, CCITT V.24 circuit 128, no EIA-232-D pin.</td>
</tr>
<tr>
<td><code>rsetctset</code></td>
<td>Output transmitter signal element timing (DCE source) on receiver signal element timing (DTE source) lead, CCITT V.24 circuit 128, no EIA-232-D pin.</td>
</tr>
</tbody>
</table>
Control Assignments

**rsetcrset**  
Output receiver signal element timing (DCE source) on receiver signal element timing (DTE source) lead, CCITT V.24 circuit 128, no EIA-232-D pin.

**control-character c**  
Set control-character to c, where:

- control-character  
  - is ctab, discard, dsusp, eof, eol, eol2, erase, intr, kill, lnext, quit, reprint, start, stop, susp, swtch, or werase (ctab is used with -stappl, see termio(7I)).

- c  
  - If c is a single character, the control character will be set to that character.

  In the POSIX locale, if c is preceded by a caret (^) indicating an escape from the shell and is one of those listed in the ^c column of the following table, then its value used (in the Value column) is the corresponding control character (for example, "^d" is a CTRL-D). "^?" is interpreted as DEL and "^-" is interpreted as undefined.

<table>
<thead>
<tr>
<th><code>c</code></th>
<th>Value</th>
<th><code>c</code></th>
<th>Value</th>
<th><code>c</code></th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>a, A</td>
<td>&lt;SOH&gt;</td>
<td>l, L</td>
<td>&lt;FF&gt;</td>
<td>w, W</td>
<td>&lt;ETB&gt;</td>
</tr>
<tr>
<td>b, B</td>
<td>&lt;STX&gt;</td>
<td>m, M</td>
<td>&lt;CR&gt;</td>
<td>x, X</td>
<td>&lt;CAN&gt;</td>
</tr>
<tr>
<td>c, C</td>
<td>&lt;ETX&gt;</td>
<td>n, N</td>
<td>&lt;SO&gt;</td>
<td>y, Y</td>
<td>&lt;EM&gt;</td>
</tr>
<tr>
<td>d, D</td>
<td>&lt;EOT&gt;</td>
<td>o, O</td>
<td>&lt;SI&gt;</td>
<td>z, Z</td>
<td>&lt;SUB&gt;</td>
</tr>
<tr>
<td>e, E</td>
<td>&lt;ENQ&gt;</td>
<td>p, P</td>
<td>&lt;DLE&gt;</td>
<td>\</td>
<td>&lt;FS&gt;</td>
</tr>
<tr>
<td>f, F</td>
<td>&lt;ACK&gt;</td>
<td>q, Q</td>
<td>&lt;DC1&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>g, G</td>
<td>&lt;BEL&gt;</td>
<td>r, R</td>
<td>&lt;DC2&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>h, H</td>
<td>&lt;BS&gt;</td>
<td>s, S</td>
<td>&lt;DC3&gt;</td>
<td>^</td>
<td>&lt;US&gt;</td>
</tr>
<tr>
<td>i, I</td>
<td>&lt;HT&gt;</td>
<td>t, T</td>
<td>&lt;DC4&gt;</td>
<td>_</td>
<td>&lt;DEL&gt;</td>
</tr>
<tr>
<td>j, J</td>
<td>&lt;LF&gt;</td>
<td>u, U</td>
<td>&lt;NAK&gt;</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>k, K</td>
<td>&lt;VT&gt;</td>
<td>v, V</td>
<td>&lt;SYN&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**min number**  
**time number**  
Set the value of min or time to number. MIN and TIME are used in Non-Canonical mode input processing (-icanon).
Set line discipline to \( i \) \((0 < i < 127)\).

Set the current terminal characteristics to the saved settings produced by the \(-g\) option.

Enable \( \text{parenb} \) and \( \text{cs7} \), or disable \( \text{parodd} \).

Enable \( \text{parenb} \), \( \text{cs7} \), and \( \text{parodd} \).

Enable \( \text{parenb} \), \( \text{cs7} \), and \( \text{parext} \).

Enable \( \text{parenb} \), \( \text{cs7} \), \( \text{parodd} \), and \( \text{parext} \).

Disable \( \text{parenb} \), and set \( \text{cs8} \).

Disable \( \text{parenb} \) and \( \text{parodd} \), and set \( \text{cs8} \).

Disable \( \text{parenb} \) and \( \text{parext} \), and set \( \text{cs8} \).

Disable \( \text{parenb} \), \( \text{parodd} \), and \( \text{parext} \), and set \( \text{cs8} \).

Enable (disable) raw input and output. Raw mode is equivalent to setting:

\[
\text{stty} \ \text{cs8} \ \text{-icanon} \ 1 \ \text{time} \ 0 \ \text{-isig} \ \text{-xcase} \ \text{\textbackslash} \\
\text{-inpck} \ \text{-opost}
\]

Unset (set) \( \text{icrnl} \), \( \text{onlcr} \). In addition \(-\text{nl}\) unsets \( \text{inlcr} \), \( \text{igncr} \), \( \text{ocrnl} \), and \( \text{onlret} \).

Set (unset) \( \text{icrnl} \). In addition, \(-\text{nl}\) unsets \( \text{inlcr} \), \( \text{igncr} \), \( \text{ocrnl} \), and \( \text{onlret} \); \(-\text{nl}\) sets \( \text{onlcr} \), and \( \text{nl} \) unsets \( \text{onlcr} \).

Set (unset) \( \text{xcase} \), \( \text{iuclc} \), and \( \text{olcuc} \).

Same as \text{lcase} \(-\text{lcase}\).

Preserve (expand to spaces) tabs when printing.

Reset \( \text{ERASE} \) and \( \text{KILL} \) characters back to normal \# and \@.

Resets all modes to some reasonable values.

Set all modes suitable for the terminal type \textit{term}, where \textit{term} is one of \texttt{tty33}, \texttt{ttyp37}, \texttt{vt05}, \texttt{tn300}, \texttt{ti700}, or \texttt{tek}.

Set normal asynchronous communications where clock settings are \texttt{xcibrg}, \texttt{rcibrg}, \texttt{tsetcoff} and \texttt{rsetcoff}.

Set window size to \( n \) rows.

Set window size to \( n \) columns.

\textbf{Combination Modes}

\begin{tabular}{|l|l|}
\hline
\textbf{line} \textit{i} & Set line discipline to \( i \) \((0 < i < 127)\). \\
\hline
\textit{saved settings} & Set the current terminal characteristics to the saved settings produced by the \texttt{-g} option. \\
\hline
\textit{evenp or parity} & Enable \texttt{parenb} and \texttt{cs7}, or disable \texttt{parodd}. \\
\hline
\textit{oddp} & Enable \texttt{parenb}, \texttt{cs7}, and \texttt{parodd}. \\
\hline
\textit{spacep} & Enable \texttt{parenb}, \texttt{cs7}, and \texttt{parext}. \\
\hline
\textit{markp} & Enable \texttt{parenb}, \texttt{cs7}, \texttt{parodd}, and \texttt{parext}. \\
\hline
\textit{-parity, or -evenp} & Disable \texttt{parenb}, and set \texttt{cs8}. \\
\hline
\textit{-oddp} & Disable \texttt{parenb} and \texttt{parodd}, and set \texttt{cs8}. \\
\hline
\textit{-spacep} & Disable \texttt{parenb} and \texttt{parext}, and set \texttt{cs8}. \\
\hline
\textit{-markp} & Disable \texttt{parenb}, \texttt{parodd}, \texttt{parext}, and set \texttt{cs8}. \\
\hline
\textit{raw (-raw or cooked)} & Enable (disable) raw input and output. Raw mode is equivalent to setting:
\end{tabular}

\texttt{stty} \ \texttt{cs8} \ \texttt{-icanon} \ 1 \ \texttt{time} \ 0 \ \texttt{-isig} \ \texttt{-xcase} \ \texttt{\textbackslash} \\
\texttt{-inpck} \ \texttt{-opost}

\textbf{User Commands}
cols \( n \) Set window size to \( n \) columns. Note that cols is a shorthand alias for columns.

ypixels \( n \) Set vertical window size to \( n \) pixels.

xpixels \( n \) Set horizontal window size to \( n \) pixels.

### USAGE

The -g flag is designed to facilitate the saving and restoring of terminal state from the shell level. For example, a program may:

```bash
saveterm="$(stty -g)" # save terminal state
stty (new settings) # set new state
... # ...
stty $saveterm # restore terminal state
```

Since the -a format is so loosely specified, scripts that save and restore terminal settings should use the -g option.

### ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of stty: LC_CTYPE, LC_MESSAGES, and NLSPATH.

### EXIT STATUS

The following exit values are returned:

- 0 Successful completion.
- >0 An error occurred.

### ATTRIBUTES

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

<table>
<thead>
<tr>
<th>/usr/bin/stty</th>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>/usr/xpg4/bin/stty</th>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWxcu4</td>
<td></td>
</tr>
</tbody>
</table>

### SEE ALSO

tabs(1), ioctl(2), write(2), getwidth(3C), attributes(5), environ(5), ldterm(7M), termio(7I), termiox(7I)
**NAME**

stty — set the options for a terminal

**SYNOPSIS**

```
/usr/ucb/stty [-a] [-g] [-h] [modes]
```

**DESCRIPTION**

stty sets certain terminal I/O options for the device that is the current standard output; without arguments, it reports the settings of certain options.

In this report, if a character is preceded by a caret (^), then the value of that option is the corresponding CTRL character (for example, "'^h" is CTRL-H; in this case, recall that CTRL-H is the same as the "back-space" key.) The sequence "'^;" means that an option has a null value.

- `-a`
  Report all of the option settings.

- `-g`
  Report current settings in a form that can be used as an argument to another stty command.

- `-h`
  Report all the option settings with the control characters in an easy to read column format.

Options in the last group are implemented using options in the previous groups. Note: Many combinations of options make no sense, but no sanity checking is performed. Hardware flow control and clock modes options may not be supported by all hardware interfaces. The options are selected from the following:

**Special Requests**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>Reports the same option settings as stty without arguments, but with the control characters in column format.</td>
</tr>
<tr>
<td>everything</td>
<td>Everything stty knows about is printed. Same as -h option.</td>
</tr>
<tr>
<td>speed</td>
<td>The terminal speed alone is reported on the standard output.</td>
</tr>
<tr>
<td>size</td>
<td>The terminal (window) sizes are printed on the standard output, first rows and then columns. This option is only appropriate if currently running a window system.</td>
</tr>
</tbody>
</table>

`size` and `speed` always report on the settings of `/dev/tty`, and always report the settings to the standard output.

**Control Modes**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parenb (-parenb)</td>
<td>Enable (disable) parity generation and detection.</td>
</tr>
<tr>
<td>parext (-parext)</td>
<td>Enable (disable) extended parity generation and detection for mark and space parity.</td>
</tr>
<tr>
<td>parodd (-parodd)</td>
<td>Select odd (even) parity, or mark (space) parity if parext is enabled.</td>
</tr>
<tr>
<td>cs5 cs6 cs7 cs8</td>
<td>Select character size (see termio(7)).</td>
</tr>
<tr>
<td>0</td>
<td>Hang up line immediately.</td>
</tr>
</tbody>
</table>
Set terminal baud rate to the number given, if possible. (All speeds are not supported by all hardware interfaces.)

**ispeed** 0 110 300 600 1200 1800 2400 4800 9600 19200 exta 38400 extb
Set terminal input baud rate to the number given, if possible. (Not all hardware supports split baud rates.) If the input baud rate is set to zero, the input baud rate will be specified by the value of the output baud rate.

**ospeed** 0 110 300 600 1200 1800 2400 4800 9600 19200 exta 38400 extb
Set terminal output baud rate to the number given, if possible. (Not all hardware supports split baud rates.) If the baud rate is set to zero, the line will be hung up immediately.

**hupcl** (-hupcl)
Hang up (do not hang up) connection on last close.

**hup** (-hup)
Same as hupcl (-hupcl).

**cstopb** (-cstopb)
Use two (one) stop bits per character.

**cread** (-cread)
Enable (disable) the receiver.

**clocal** (-clocal)
Assume a line without (with) modem control.

**crtscts** (-crtscts)
Enable hardware flow control. Raise the RTS (Request to Send) modem control line. Suspends output until the CTS (Clear to Send) line is raised.

**loblk** (-loblk)
Block (do not block) output from a non-current layer.

**Input Modes**

**ignbrk** (-ignbrk)
Ignore (do not ignore) break on input.

**brkint** (-brkint)
Signal (do not signal) INTR on break.

**ignpar** (-ignpar)
Ignore (do not ignore) parity errors.

**parmrk** (-parmrk)
Mark (do not mark) parity errors (see termio(7I)).

**inpck** (-inpck)
Enable (disable) input parity checking.

**istrip** (-istrip)
Strip (do not strip) input characters to seven bits.

**inlcr** (-inlcr)
Map (do not map) NL to CR on input.

**igncr** (-igncr)
Ignore (do not ignore) CR on input.

**icrnl** (-icrnl)
Map (do not map) CR to NL on input.
iuclc (-iuclc) Map (do not map) upper-case alphabetics to lower case on input.

ixon (-ixon) Enable (disable) START/STOP output control. Output is stopped by sending an STOP and started by sending an START.

ixany (-ixany) Allow any character (only START) to restart output.

decctlq (-decctlq) Same as -ixany.

ixoff (-ixoff) Request that the system send (not send) START/STOP characters when the input queue is nearly empty/full.

tandem (-tandem) Same as ixoff.

imaxbel (-imaxbel) Echo (do not echo) BEL when the input line is too long.

iexten (-iexten) Enable (disable) extended (implementation-defined) functions for input data.

opost (-opost) Post-process output (do not post-process output; ignore all other output modes).

olcuc (-olcuc) Map (do not map) lower-case alphabetics to upper case on output.

onlcr (-onlcr) Map (do not map) NL to CR-NL on output.

ocrnl (-ocrnl) Map (do not map) CR to NL on output.

onocr (-onocr) Do not (do) output CRs at column zero.

onlret (-onlret) On the terminal NL performs (does not perform) the CR function.

ofill (-ofill) Use fill characters (use timing) for delays.

ofdel (-ofdel) Fill characters are DELs (NULs).

cr0 cr1 cr2 cr3 Select style of delay for carriage returns (see termio(7I)).

nl0 nl1 Select style of delay for line-feeds (see termio(7I)).

tab0 tab1 tab2 tab3 Select style of delay for horizontal tabs (see termio(7I)).

bs0 bs1 Select style of delay for backspaces (see termio(7I)).

ff0 ff1 Select style of delay for form-feeds (see termio(7I)).

vt0 vt1 Select style of delay for vertical tabs (see termio(7I)).

Local Modes

isig (-isig) Enable (disable) the checking of characters against the special control characters INTR, QUIT, and SWITCH.
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>icanon(-icanon)</td>
<td>Enable (disable) canonical input (ERASE and KILL processing). Does not set MIN or TIME.</td>
</tr>
<tr>
<td>cbreak(-cbreak)</td>
<td>Equivalent to -icanon min 1 time 0.</td>
</tr>
<tr>
<td>xcase(-xcase)</td>
<td>Canonical (unprocessed) upper/lower-case presentation.</td>
</tr>
<tr>
<td>echo(-echo)</td>
<td>Echo back (do not echo back) every character typed.</td>
</tr>
<tr>
<td>echoe(-echoe)</td>
<td>Echo (do not echo) ERASE character as a backspace-space-backspace string. Note: This mode will erase the ERASEed character on many CRT terminals; however, it does not keep track of column position and, as a result, may be confusing on escaped characters, tabs, and backspaces.</td>
</tr>
<tr>
<td>crterase(-crterase)</td>
<td>Same as echoe.</td>
</tr>
<tr>
<td>echok(-echok)</td>
<td>Echo (do not echo) NL after KILL character.</td>
</tr>
<tr>
<td>lfkc(-lfkc)</td>
<td>The same as echok (-echok); obsolete.</td>
</tr>
<tr>
<td>echonl(-echonl)</td>
<td>Echo (do not echo) NL.</td>
</tr>
<tr>
<td>noflash(-noflash)</td>
<td>Disable (enable) flush after INTR, QUIT, or SWITCH.</td>
</tr>
<tr>
<td>stwrap(-stwrap)</td>
<td>Disable (enable) truncation of lines longer than 79 characters on a synchronous line. (Does not apply to the 3B2.)</td>
</tr>
<tr>
<td>tostop(-tostop)</td>
<td>Send (do not send) SIGTTOU for background processes.</td>
</tr>
<tr>
<td>echoctl(-echoctl)</td>
<td>Echo (do not echo) control characters as ^char, delete as ^?</td>
</tr>
<tr>
<td>ctlecho(-ctlecho)</td>
<td>Same as echoctl.</td>
</tr>
<tr>
<td>echoprt(-echoprt)</td>
<td>Echo (do not echo) erase character as character is “erased”.</td>
</tr>
<tr>
<td>prterase(-prterase)</td>
<td>Same as echoprt.</td>
</tr>
<tr>
<td>echoke(-echoke)</td>
<td>BS-SP-BS erase (do not BS-SP-BS erase) entire line on line kill.</td>
</tr>
<tr>
<td>crtkill(-crtkill)</td>
<td>Same as echoke.</td>
</tr>
<tr>
<td>flusho(-flusho)</td>
<td>Output is (is not) being flushed.</td>
</tr>
<tr>
<td>pendin(-pendin)</td>
<td>Retype (do not retype) pending input at next read or input character.</td>
</tr>
<tr>
<td>stflush(-stflush)</td>
<td>Enable (disable) flush on a synchronous line after every write(2). (Does not apply to the 3B2.)</td>
</tr>
</tbody>
</table>
### Hardware Flow Control Modes

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stapl (-stappl)</td>
<td>Use application mode (use line mode) on a synchronous line. (Does not apply to the 3B2.)</td>
</tr>
<tr>
<td>rtsxoff (-rtsxoff)</td>
<td>Enable (disable) RTS hardware flow control on input.</td>
</tr>
<tr>
<td>ctsxon (-ctxon)</td>
<td>Enable (disable) CTS hardware flow control on output.</td>
</tr>
<tr>
<td>dterxoff (-dterxoff)</td>
<td>Enable (disable) DTER hardware flow control on input.</td>
</tr>
<tr>
<td>rlsdxon (-rlsdxon)</td>
<td>Enable (disable) RLSD hardware flow control on output.</td>
</tr>
<tr>
<td>isxoff (-isxoff)</td>
<td>Enable (disable) isochronous hardware flow control on input.</td>
</tr>
</tbody>
</table>

### Clock Modes

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xcibrg</td>
<td>Get transmit clock from internal baud rate generator.</td>
</tr>
<tr>
<td>xctset</td>
<td>Get the transmit clock from transmitter signal element timing (DCE source) lead, CCITT V.24 circuit 114, EIA-232-D pin 15.</td>
</tr>
<tr>
<td>xcrset</td>
<td>Get transmit clock from receiver signal element timing (DCE source) lead, CCITT V.24 circuit 115, EIA-232-D pin 17.</td>
</tr>
<tr>
<td>rcibrg</td>
<td>Get receive clock from internal baud rate generator.</td>
</tr>
<tr>
<td>rctset</td>
<td>Get receive clock from transmitter signal element timing (DCE source) lead, CCITT V.24 circuit 114, EIA-232-D pin 15.</td>
</tr>
<tr>
<td>rcrset</td>
<td>Get receive clock from receiver signal element timing (DCE source) lead, CCITT V.24 circuit 115, EIA-232-D pin 17.</td>
</tr>
<tr>
<td>tsetcoff</td>
<td>Transmitter signal element timing clock not provided.</td>
</tr>
<tr>
<td>tsetcrc</td>
<td>Output receive clock on transmitter signal element timing (DTE source) lead, CCITT V.24 circuit 113, EIA-232-D pin 24, clock source.</td>
</tr>
<tr>
<td>tsetcxc</td>
<td>Output transmit clock on transmitter signal element timing (DTE source) lead, CCITT V.24 circuit 113, EIA-232-D pin 24, clock source.</td>
</tr>
<tr>
<td>rsetcoff</td>
<td>Receiver signal element timing clock not provided.</td>
</tr>
<tr>
<td>rsetcrc</td>
<td>Output receive clock on receiver signal element timing (DTE source) lead, CCITT V.24 circuit 128, no EIA-232-D pin, clock source.</td>
</tr>
<tr>
<td>rsetcxc</td>
<td>Output transmit clock on receiver signal element timing (DTE source) lead, CCITT V.24 circuit 128, no EIA-232-D pin, clock source.</td>
</tr>
</tbody>
</table>
### Control Assignments

| control-character $c$ | Set control-character to $c$, where control-character is intr, quit, erase, kill, eof, eol, eol2, swtch, start, stop, susp, dsusp, rprnt, flush, werase, lnext min, ctab, time, or brk (ctab is used with -stappl; min and time are used with -icanon; see termios(7I)). If $c$ is preceded by an (escaped from the shell) caret (^), then the value used is the corresponding CTRL character (for example, “^d” is a CTRL-d); “^?” is interpreted as DEL and “^-” is interpreted as undefined. |

| `line i` | Set line discipline to $i$ ($0 < i < 127$). |

### Combination Modes

| evenp or parity | Enable parenb and cs7. |
| -evenp, or -parity | Disable parenb, and set cs8. |
| even (-even) | Same as evenp (-evenp). |
| oddp | Enable parenb, cs7, and parodd. |
| -oddp | Disable parenb and parodd, and set cs8. |
| odd (-odd) | Same as oddp (-oddp). |
| spacep | Enable parenb, cs7, and parext. |
| -spacep | Disable parenb and parext, and set cs8. |
| markp | Enable parenb, cs7, parodd, and parext. |
| -markp | Disable parenb, parodd, and parext, and set cs8. |
| raw (-raw or cooked) | Enable (disable) raw input and output (no ERASE, KILL, INTR, SWTCH, EOT, or output post processing). |
| nl (-nl) | Unset (set) icrnl, onlcr. In addition -nl unsets inlcr, igncr, ocrnl, and onlret. |
| lcase (-lcase) | Set (unset) xcase, iuc1c, and olcuc. |
| LCASE (-LCASE) | Same as lcase (-lcase). |
| tabs (-tabs or tab3) | Preserve (expand to spaces) tabs when printing. |
| ek | Reset ERASE and KILL characters back to normal # and @. |
| sane | Resets all modes to some reasonable values. |
| term | Set all modes suitable for the terminal type term, where term is one of tty33, tty37, vt05, tn300, ti700, or tek. |
async

Set normal asynchronous communications where clock settings are xcibrg, rcibrg, tsetcoff and rsetcoff.

litout (-litout)

Disable (enable) parenb, istrip, and opost, and set cs8 (cs7).

pass8 (-pass8)

Disable (enable) parenb and istrip, and set cs8 (cs7).

crt

Set options for a CRT (echoe, echoctl, and, if >= 1200 baud, echoke.)

dec

Set all modes suitable for Digital Equipment Corp. operating systems users ERASE, KILL, and INTR characters to ^?, ^U, and ^C, decctlq, and crt.)

Window Size

rowsn

Set window size to n rows.

columnsn

Set window size to n columns.

colsn

An alias for columns n.

ypixelsn

Set vertical window size to n pixels.

xpixelsn

Set horizontal window size to n pixels.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

SEE ALSO

tabs(1), ioctl(2), attributes(5), termio(7I), termiox(7I)
NAME

sum – print checksum and block count for a file

SYNOPSIS

csum [-r] [file...]

DESCRIPTION

The sum utility calculates and prints a 16-bit checksum for the named file and the number of 512-byte blocks in the file. It is typically used to look for bad spots, or to validate a file communicated over some transmission line.

OPTIONS

The following options are supported:

- Use an alternate (machine-dependent) algorithm in computing the checksum.

OPERANDS

The following operands are supported:

file A path name of a file. If no files are named, the standard input is used.

USAGE

See largefile(5) for the description of the behavior of sum when encountering files greater than or equal to 2 Gbyte (2^31 bytes).

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of sum: LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS

The following exit values are returned.

0 Successful completion.

>0 An error occurred.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

SEE ALSO

csum(1), sum(1B), wc(1), attributes(5), environ(5), largefile(5)

DIAGNOSTICS

“Read error” is indistinguishable from end of file on most devices; check the block count.

NOTES

Portable applications should use csum(1).

sum and usr/ucb/sum (see sum(1B)) return different checksums.
NAME | sum – calculate a checksum for a file

SYNOPSIS | /usr/ucb/sum file...

DESCRIPTION | sum calculates and displays a 16-bit checksum for the named file and displays the size of the file in kilobytes. It is typically used to look for bad spots, or to validate a file communicated over some transmission line. The checksum is calculated by an algorithm which may yield different results on machines with 16-bit ints and machines with 32-bit ints, so it cannot always be used to validate that a file has been transferred between machines with different-sized ints.

USAGE | See largefile(5) for the description of the behavior of sum when encountering files greater than or equal to 2 Gbyte (2^31 bytes).

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

SEE ALSO | sum(1), wc(1), attributes(5), largefile(5)

DIAGNOSTICS | Read error is indistinguishable from EOF on most devices; check the block count.

NOTES | sum and /usr/bin/sum (see sum(1)) return different checksums.

This utility is obsolete.
suspend(1)

NAME    suspend – shell built-in function to halt the current shell

sh    suspend

csh    suspend

ksh    suspend

sh    Stops the execution of the current shell (but not if it is the login shell).

csh    Stop the shell in its tracks, much as if it had been sent a stop signal with ^z. This is most often used to stop shells started by su.

ksh    Stops the execution of the current shell (but not if it is the login shell).

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO    csh(1), kill(1), ksh(1), sh(1), su(1M), attributes(5)
NAME symorder – rearrange a list of symbols
SYNOPSIS symorder [-s] objectfile symbolfile
DESCRIPTION symorder was used in SunOS 4.x specifically to cut down on the overhead of getting symbols from vmunix. This is no longer applicable as kernel symbol entries are dynamically obtained through /dev/ksyms.

This script is provided as a convenience for software developers who need to maintain scripts that are portable across a variety of operating systems.

EXIT STATUS symorder has exit status 0.
ATTRIBUTES See attributes(5) for descriptions of the following attributes:

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWbtool</td>
</tr>
</tbody>
</table>

SEE ALSO nlist(3ELF), attributes(5), ksym(7D).
sysV-make – maintain, update, and regenerate groups of programs

SYNOPSIS
/usr/lib/svr4.make [-f makefile] [-eiknpqrst] [names]

DESCRIPTION
This is the “vanilla” System V version of make. If the environment variable
USE_SVR4_MAKE is set, then the command make will invoke this version of make.
(See also the ENVIRONMENT section.)

make allows the programmer to maintain, update, and regenerate groups of computer
programs. make executes commands in makefile to update one or more target names
(names are typically programs). If the -f option is not present, then makefile,
Makefile, and the Source Code Control System (SCCS) files s.makefile and
s.Makefile are tried in order. If makefile is ‘−’ the standard input is taken. More than
one -f makefile argument pair may appear.

make updates a target only if its dependents are newer than the target. All prerequisite
files of a target are added recursively to the list of targets. Missing files are deemed to
be outdated.

The following list of four directives can be included in makefile to extend the options
provided by make. They are used in makefile as if they were targets:

<table>
<thead>
<tr>
<th>Directive</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.DEFAULT:</td>
<td>If a file must be made but there are no explicit commands or relevant built-in rules, the commands associated with the name .DEFAULT are used if it exists.</td>
</tr>
<tr>
<td>.IGNORE:</td>
<td>Same effect as the -i option.</td>
</tr>
<tr>
<td>.PRECIOUS:</td>
<td>Dependents of the .PRECIOUS entry will not be removed when quit or interrupt are hit.</td>
</tr>
<tr>
<td>.SILENT:</td>
<td>Same effect as the -s option.</td>
</tr>
</tbody>
</table>

Options
The options for make are listed below:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-e</td>
<td>Environment variables override assignments within makefiles.</td>
</tr>
<tr>
<td>-f makefile</td>
<td>Description filename (makefile is assumed to be the name of a description file).</td>
</tr>
<tr>
<td>-i</td>
<td>Ignore error codes returned by invoked commands.</td>
</tr>
<tr>
<td>-k</td>
<td>Abandon work on the current entry if it fails, but continue on other branches that do not depend on that entry.</td>
</tr>
<tr>
<td>-n</td>
<td>No execute mode. Print commands, but do not execute them. Even command lines beginning with an ‘@’ are printed.</td>
</tr>
<tr>
<td>-p</td>
<td>Print out the complete set of macro definitions and target descriptions.</td>
</tr>
<tr>
<td>-q</td>
<td>Question. make returns a zero or non-zero status code depending on whether or not the target file has been updated.</td>
</tr>
<tr>
<td>-r</td>
<td>Do not use the built-in rules.</td>
</tr>
</tbody>
</table>
-s  Silent mode. Do not print command lines before executing.
-t  Touch the target files (causing them to be updated) rather than issue the usual commands.

The makefile invoked with the `-f` option is a carefully structured file of explicit instructions for updating and regenerating programs, and contains a sequence of entries that specify dependencies. The first line of an entry is a blank-separated, non-null list of targets, then a `;`, then a (possibly null) list of prerequisite files or dependencies. Text following a `;` and all following lines that begin with a tab are shell commands to be executed to update the target. The first non-empty line that does not begin with a tab or `#` begins a new dependency or macro definition. Shell commands may be continued across lines with a backslash-new-line (`\-NEWLINE`) sequence. Everything printed by make (except the initial TAB) is passed directly to the shell as is. Thus,

```plaintext
echo a\b
```

will produce

```
ab
```

exactly the same as the shell would.

Number-sign (#) and NEWLINE surround comments including contained `\-NEWLINE` sequences.

The following makefile says that `pgm` depends on two files `a.o` and `b.o`, and that they in turn depend on their corresponding source files (`a.c` and `b.c`) and a common file `incl.h`:

```plaintext
pgm:  a.o  b.o
     cc  a.o  b.o  -o  pgm
a.o: incl.h  a.c
     cc  -c  a.c
b.o: incl.h  b.c
     cc  -c  b.c
```

Command lines are executed one at a time, each by its own shell. The `SHELL` environment variable can be used to specify which shell make should use to execute commands. The default is `/usr/bin/sh`. The first one or two characters in a command can be the following: `@`, `−`, `@−`, or `−@`. If `@` is present, printing of the command is suppressed. If `−` is present, make ignores an error. A line is printed when it is executed unless the `-s` option is present, or the entry `.SILENT:` is included in `makefile`, or unless the initial character sequence contains a `@`. The `-n` option specifies
printing without execution; however, if the command line has the string `$(MAKE)` in it, the line is always executed (see the discussion of the `MAKEFLAGS` macro in the make Environment sub-section below). The `-t` (touch) option updates the modified date of a file without executing any commands.

Commands returning non-zero status normally terminate `make`. If the `-i` option is present, if the entry `.IGNORE:` is included in `makefile`, or if the initial character sequence of the command contains `'-'`, the error is ignored. If the `-k` option is present, work is abandoned on the current entry, but continues on other branches that do not depend on that entry.

Interrupt and quit cause the target to be deleted unless the target is a dependent of the directive `.PRECIOUS`.

The environment is read by `make`. All variables are assumed to be macro definitions and are processed as such. The environment variables are processed before any `makefile` and after the internal rules; thus, macro assignments in a `makefile` override environment variables. The `-e` option causes the environment to override the macro assignments in a `makefile`. Suffixes and their associated rules in the `makefile` will override any identical suffixes in the built-in rules.

The `MAKEFLAGS` environment variable is processed by `make` as containing any legal input option (except `-f` and `-p`) defined for the command line. Further, upon invocation, `make` “invents” the variable if it is not in the environment, puts the current options into it, and passes it on to invocations of commands. Thus, `MAKEFLAGS` always contains the current input options. This feature proves very useful for “super-makes”. In fact, as noted above, when the `-n` option is used, the command `$(MAKE)` is executed anyway; hence, one can perform a `make -n` recursively on a whole software system to see what would have been executed. This result is possible because the `-n` is put in `MAKEFLAGS` and passed to further invocations of `$(MAKE)`. This usage is one way of debugging all of the `makefiles` for a software project without actually doing anything.

If the string `include` appears as the first seven letters of a line in a `makefile`, and is followed by a blank or a tab, the rest of the line is assumed to be a filename and will be read by the current invocation, after substituting for any macros.

Entries of the form `string1 = string2` are macro definitions. `string2` is defined as all characters up to a comment character or an unescaped NEWLINE. Subsequent appearances of `$(string1[:subst1=[subst2]])` are replaced by `string2`. The parentheses are optional if a single-character macro name is used and there is no substitute sequence. The optional `:subst1=subst2` is a substitute sequence. If it is specified, all non-overlapping occurrences of `subst1` in the named macro are replaced by `subst2`. Strings (for the purposes of this type of substitution) are delimited by BLANKs, TABs, NEWLINE characters, and beginnings of lines. An example of the use of the substitute sequence is shown in the Libraries sub-section below.
Internal Macros

There are five internally maintained macros that are useful for writing rules for building targets.

$*
The macro $* stands for the filename part of the current dependent with the suffix deleted. It is evaluated only for inference rules.

$@
The $@ macro stands for the full target name of the current target. It is evaluated only for explicitly named dependencies.

$<The $< macro is only evaluated for inference rules or the .DEFAULT rule. It is the module that is outdated with respect to the target (the “manufactured” dependent file name). Thus, in the .c.o rule, the $< macro would evaluate to the .c file. An example for making optimized .o files from .c files is:

```
c.o:
  cc -c -o $*.c
```

or:

```
c.o:
  cc -c -o $<
```

$?The $? macro is evaluated when explicit rules from the makefile are evaluated. It is the list of prerequisites that are outdated with respect to the target, and essentially those modules that must be rebuilt.

$%The $% macro is only evaluated when the target is an archive library member of the form lib(file.o). In this case, $% evaluates to lib and $% evaluates to the library member, file.o.

Four of the five macros can have alternative forms. When an upper case D or F is appended to any of the four macros, the meaning is changed to “directory part” for D and “file part” for F. Thus, $(@D) refers to the directory part of the string $@. If there is no directory part, ./ is generated. The only macro excluded from this alternative form is $?.

Suffixes

Certain names (for instance, those ending with .o) have inferable prerequisites such as .c, .s, etc. If no update commands for such a file appear in makefile, and if an inferable prerequisite exists, that prerequisite is compiled to make the target. In this case, make has inference rules that allow building files from other files by examining the suffixes and determining an appropriate inference rule to use. The current default inference rules are:

```
.c .c~ .f .f~ .s .s~ .sh .sh~ .C .C~
.c.a .c.o .c~.a .c~.c .c~.o .f.a .f.o .f~.a .f~.f .f~.o
.h~.h l.c l.o l~.c l~.l l~.o s.a s.o s~.a s~.o
```

User Commands 1479
The internal rules for `make` are contained in the source file `make.rules` for the `make` program. These rules can be locally modified. To print out the rules compiled into the `make` on any machine in a form suitable for recompilation, the following command is used:

```
make -pf - 2>/dev/null </dev/null
```

A tilde in the above rules refers to an SCCS file (see `sccsfile(4)`). Thus, the rule `.c~.o` would transform an SCCS C source file into an object file (.o). Because the `s.` of the SCCS files is a prefix, it is incompatible with the `make` suffix point of view. Hence, the tilde is a way of changing any file reference into an SCCS file reference.

A rule with only one suffix (for example, `.c:`) is the definition of how to build `x` from `x.c`. In effect, the other suffix is null. This feature is useful for building targets from only one source file, for example, shell procedures and simple C programs.

Additional suffixes are given as the dependency list for `.SUFFIXES`. Order is significant: the first possible name for which both a file and a rule exist is inferred as a prerequisite. The default list is:

```
.SUFFIXES: .o .c .c~ .y .y~ .l .l~ .s .s~ .sh .sh~ .h .h~ .f .f~ .C .C~ .Y .Y~ .L .L~
```

Here again, the above command for printing the internal rules will display the list of suffixes implemented on the current machine. Multiple suffix lists accumulate; `.SUFFIXES:` with no dependencies clears the list of suffixes.

### Inference Rules

The first example can be done more briefly.

```
pgm: a.o b.o
   cc a.o b.o -o pgm
   a.o b.o: incl.h
```

This abbreviation is possible because `make` has a set of internal rules for building files. The user may add rules to this list by simply putting them in the `makefile`.

Certain macros are used by the default inference rules to permit the inclusion of optional matter in any resulting commands. For example, `CFLAGS`, `LFLAGS`, and `YFLAGS` are used for compiler options to `cc(1B)`. Again, the previous method for examining the current rules is recommended.
The inference of prerequisites can be controlled. The rule to create a file with suffix `.o` from a file with suffix `.c` is specified as an entry with `.c.o` as the target and no dependents. Shell commands associated with the target define the rule for making a `.o` file from a `.c` file. Any target that has no slashes in it and starts with a dot is identified as a rule and not a true target.

**Libraries**

If a target or dependency name contains parentheses, it is assumed to be an archive library, the string within parentheses referring to a member within the library. Thus, `lib(file.o)` and `${LIB}(file.o)` both refer to an archive library that contains `file.o`. (This example assumes the LIB macro has been previously defined.) The expression `${LIB}[file1.o file2.o]` is not legal. Rules pertaining to archive libraries have the form `.XX.a` where the XX is the suffix from which the archive member is to be made. An unfortunate by-product of the current implementation requires the XX to be different from the suffix of the archive member. Thus, one cannot have `lib(file.o)` depend upon `file.o` explicitly. The most common use of the archive interface follows. Here, we assume the source files are all C type source:

```
lib: lib(file1.o) lib(file2.o) lib(file3.o)
    @echo lib is now up-to-date
    .c.a:
    $(CC) -c $(CFLAGS) $<
    $(AR) $(ARFLAGS) $@ $*.o
    rm -f $*.o
```

In fact, the `.c.a` rule listed above is built into make and is unnecessary in this example. A more interesting, but more limited example of an archive library maintenance construction follows:

```
lib: lib(file1.o) lib(file2.o) lib(file3.o)
    $(CC) -c $(CFLAGS) $(?:.o=.c)
    $(AR) $(ARFLAGS) lib $?
    rm $?
    @echo lib is now up-to-date
    .c.a:
```

Here the substitution mode of the macro expansions is used. The `?` list is defined to be the set of object filenames (inside `lib`) whose C source files are outdated. The substitution mode translates the `.o` to `.c` (Unfortunately, one cannot as yet transform to `.c-`; however, this transformation may become possible in the future.) Also note the disabling of the `.c.a:` rule, which would have created each object file, one by one. This particular construct speeds up archive library maintenance considerably. This type of construct becomes very cumbersome if the archive library contains a mix of assembly programs and C programs.

**Environment Variables**

`USE_SVR4_MAKE` If this environment variable is set, then the make command will invoke this System V version of make. If this variable is not set, then the default version of make(1S) is invoked.

`USE_SVR4_MAKE` can be set as follows (Bourne shell):

```
$ USE_SVR4_MAKE='''; export USE_SVR4_MAKE
```
or (C shell):

```
% setenv USE_SVR4_MAKE
```

**FILES**

[Mm]akefile

s. [Mm]akefile

- default makefiles

/usr/bin/sh

- default shell for make

/usr/share/lib/make/make.rules

- default rules for make

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsprot</td>
</tr>
</tbody>
</table>

**SEE ALSO**

cc(1B), cd(1), make(1S), sh(1), printf(3C), sccsfile(4), attributes(5)

**NOTES**

Some commands return non-zero status inappropriately; use -i or the `-' command line prefix to overcome the difficulty.

Filenames containing the characters ‘=’, ‘;’, and ‘@’ will not work. Commands that are directly executed by the shell, notably cd(1), are ineffectual across NEWLINEs in make. The syntax `lib(file1.o file2.o file3.o)` is illegal. You cannot build `lib(file.o)` from `file.o`. 
NAME | tabs – set tabs on a terminal
SYNOPSIS | **tabs** [-n | --file [ [-code] | -a | -a2 | -c | -c2 | -c3 | -f | -p | -s | -u]] [+m [n]] [-T type]
DESCRIPTION | **tabs** sets the tab stops on the user’s terminal according to a tab specification, after clearing any previous settings. The user’s terminal must have remotely settable hardware tabs.
OPTIONS | The following options are supported. If a given flag occurs more than once, the last value given takes effect:

- **-T type**
  tabs needs to know the type of terminal in order to set tabs and margins. *type* is a name listed in term(5). If no -T flag is supplied, tabs uses the value of the environment variable TERM. If the value of TERM is NULL or TERM is not defined in the environment (see environ(5)), tabs uses ansi+tabs as the terminal type to provide a sequence that will work for many terminals.

- **+m [n]**
  The margin argument may be used for some terminals. It causes all tabs to be moved over *n* columns by making column *n*+1 the left margin. If +m is given without a value of *n*, the value assumed is 10. For a TeraNet, the first value in the tab list should be 1, or the margin will move even further to the right. The normal (leftmost) margin on most terminals is obtained by +m0. The margin for most terminals is reset only when the +m flag is given explicitly.

Tab Specification | Four types of tab specification are accepted. They are described below: canned, repetitive (-n), arbitrary (*n1,*n2,...), and file (--file).

If no tab specification is given, the default value is -8, that is, UNIX system "standard" tabs. The lowest column number is 1. Note: For tabs, column 1 always refers to the leftmost column on a terminal, even one whose column markers begin at 0, for example, the DASI 300, DASI 300s, and DASI 450.

**Canned -code**
Use one of the codes listed below to select a canned set of tabs. If more than one code is specified, the last code option will be used. The legal codes and their meanings are as follows:

- **-a**
  1, 10, 16, 36, 72  Assembler, IBM S/370, first format

- **-a2**
  1, 10, 16, 40, 72  Assembler, IBM S/370, second format

- **-c**
  1, 8, 12, 16, 20, 55  COBOL, normal format
COBOL compact format (columns 1-6 omitted). Using this code, the first typed character corresponds to card column 7, one space gets you to column 8, and a tab reaches column 12. Files using this tab setup should include a format specification as follows (see \texttt{fspec(4)}):
\begin{verbatim}
<:t-c2 m6 s66 d:>
\end{verbatim}

COBOL compact format (columns 1-6 omitted), with more tabs than \texttt{-c2}. This is the recommended format for COBOL. The appropriate format specification is (see \texttt{fspec(4)}):
\begin{verbatim}
<:t-c3 m6 s66 d:>
\end{verbatim}

\texttt{-f} \hspace{1cm} 1,7,11,15,19,23

FORTRAN

\texttt{-p} \hspace{1cm} 1,5,9,13,17,21,25,29,33,37,41,45,49,53,57,61

PL/I

\texttt{-s} \hspace{1cm} 1,10,55

SNOBOL

\texttt{-u} \hspace{1cm} 1,12,20,44

UNIVAC 1100 Assembler

\textbf{Repetitive} \hspace{1cm} \texttt{-n} \hspace{1cm} A repetitive specification requests tabs at columns 1+n, 1+2*n, etc., where \textit{n} is a single-digit decimal number. Of particular importance is the value 8: this represents the UNIX system “standard” tab setting, and is the most likely tab setting to be found at a terminal. When \texttt{-0} is used, the tab stops are cleared and no new ones are set.

\textbf{Arbitrary} \hspace{1cm} See \texttt{OPERANDS}.

\textbf{File} \hspace{1cm} \texttt{-file} \hspace{1cm} If the name of a \texttt{file} is given, \texttt{tabs} reads the first line of the \texttt{file}, searching for a format specification (see \texttt{fspec(4)}). If it finds one there, it sets the tab stops according to it, otherwise it sets them as \texttt{-8}. This type of specification may be used to make sure that a tabbed \texttt{file} is printed with correct tab settings, and would be used with the \texttt{pr} command:
\begin{verbatim}
example\% tabs -file; pr file
\end{verbatim}

Tab and margin setting is performed via the standard output.
The following operand is supported:

\[ n1[n2,\ldots] \]

The arbitrary format consists of tab-stop values separated by commas or spaces. The tab-stop values must be positive decimal integers in ascending order. Up to 40 numbers are allowed. If any number (except the first one) is preceded by a plus sign, it is taken as an increment to be added to the previous value. Thus, the formats 1,10,20,30, and 1,10,+10,+10 are considered identical.

### Examples

**Example 1 Using the `tabs` Command**

The following command is an example using `code` (canned specification) to set tabs to the settings required by the IBM assembler: columns 1, 10, 16, 36, 72:

```bash
eexample% tabs -a
```

The next command is an example of using `-n` (repetitive specification), where \( n \) is 8, causes tabs to be set every eighth position: \( 1+(1\times8), 1+(2\times8), \ldots \) which evaluate to columns 9, 17, \ldots:

```bash
eexample% tabs -8
```

This command uses \( n1,n2,\ldots \) (arbitrary specification) to set tabs at columns 1, 8, and 36:

```bash
eexample% tabs 1,8,36
```

The last command is an example of using `-f` (file specification) to indicate that tabs should be set according to the first line of `$_HOME/fspec.list/att4425` (see `fspec(4)`).

```bash
eexample% tabs -$_HOME/fspec.list/att4425
```

### Environment Variables

See `environ(5)` for descriptions of the following environment variables that affect the execution of `tabs`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

**TERM** Determine the terminal type. If this variable is unset or null, and if the `-T` option is not specified, terminal type `ansi+tabs` will be used.

### Exit Status

The following exit values are returned:

- `0` Successful completion.
- `>0` An error occurred.

### Attributes

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>
There is no consistency among different terminals regarding ways of clearing tabs and setting the left margin.

tabs clears only 20 tabs (on terminals requiring a long sequence), but is willing to set 64.

The *tabspec* used with the `tabs` command is different from the one used with the `newform` command. For example, `tabs -8` sets every eighth position; whereas `newform -i-8` indicates that tabs are set every eighth position.
NAME
  tail – deliver the last part of a file

SYNOPSIS
  /usr/bin/tail [± number [lbcf]] [file]
  /usr/bin/tail [-lbcf] [file]
  /usr/bin/tail [± number [lbcr]] [file]
  /usr/bin/tail [-lbcr] [file]
  /usr/xpg4/bin/tail [-f | -r] [-c number | -n number] [file]
  /usr/xpg4/bin/tail [± number [l | b | c] [f]] [file]
  /usr/xpg4/bin/tail [± number [l] [f | r]] [file]

DESCRIPTION
  The tail utility copies the named file to the standard output beginning at a
  designated place. If no file is named, the standard input is used.

  Copying begins at a point in the file indicated by the -c number, -n number, or ±number
  options (if ±number is specified, begins at distance number from the beginning; if
  -number is specified, from the end of the input; if number is NULL, the value 10 is
  assumed). number is counted in units of lines or byte according to the -c or -n
  options, or lines, blocks, or bytes, according to the appended option l, b, or c. When
  no units are specified, counting is by lines.

OPTIONS
  The following options are supported for both /usr/bin/tail and
  /usr/xpg4/bin/tail. The -r and -f options are mutually exclusive. If both are
  specified on the command line, the -f option will be ignored.

  -b  Units of blocks.
  -c  Units of bytes.
  -f  Follow. If the input-file is not a pipe, the program will not terminate after
       the line of the input-file has been copied, but will enter an endless loop,
       wherein it sleeps for a second and then attempts to read and copy further
       records from the input-file. Thus it may be used to monitor the growth of a
       file that is being written by some other process.
  -l  Units of lines.
  -r  Reverse. Copies lines from the specified starting point in the file in reverse
       order. The default for r is to print the entire file in reverse order.

/usr/xpg4/bin/tail
  The following options are supported for /usr/xpg4/bin/tail only:

  -c number  The number option-argument must be a decimal integer whose
             sign affects the location in the file, measured in bytes, to begin the
             copying:
             +      Copying starts relative to the beginning of the file.
             -      Copying starts relative to the end of the file.
             none   Copying starts relative to the end of the file.
The origin for counting is 1; that is, \(-c+1\) represents the first byte of the file, \(-c-1\) the last.

\(-n\) number

Equivalent to \(-cn\umber\), except the starting location in the file is measured in lines instead of bytes. The origin for counting is 1; that is, \(-n+1\) represents the first line of the file, \(-n-1\) the last.

**OPERANDS**

The following operand is supported:

file

A path name of an input file. If no file operands are specified, the standard input will be used.

**USAGE**

See *largefile*(5) for the description of the behavior of *tail* when encountering files greater than or equal to 2 Gbyte (\(2^{31}\) bytes).

**EXAMPLES**

**EXAMPLE 1** Using the *tail* Command

The following command will print the last ten lines of the file fred, followed by any lines that are appended to fred between the time *tail* is initiated and killed.

```
example% tail -f fred
```

The next command will print the last 15 bytes of the file fred, followed by any lines that are appended to fred between the time *tail* is initiated and killed:

```
example% tail -15cf fred
```

**ENVIRONMENT VARIABLES**

See *environ*(5) for descriptions of the following environment variables that affect the execution of *tail*: LC_CTYPE, LC_MESSAGES, and NLSPATH.

**EXIT STATUS**

The following exit values are returned:

- 0 Successful completion.
- >0 An error occurred.

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>Attribute Type</th>
<th>Attribute Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute Type</th>
<th>Attribute Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWxcsu4</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>
Piped tails relative to the end of the file are stored in a buffer, and thus are limited in length. Various kinds of anomalous behavior may happen with character special files.
The talk utility is a two-way, screen-oriented communication program.

When first invoked, talk sends a message similar to:

```
Message from TalkDaemon@her_machine at time...
talk: connection requested by your_address
talk: respond with: talk your_address
```

to the specified address. At this point, the recipient of the message can reply by typing:

```
talk your_address
```

Once communication is established, the two parties can type simultaneously, with their output displayed in separate regions of the screen. Characters are processed as follows:

- Typing the alert character will alert the recipient’s terminal.
- Typing Control-L will cause the sender’s screen regions to be refreshed.
- Typing the erase and kill characters will affect the sender’s terminal in the manner described by the termios(3C) interface.
- Typing the interrupt or end-of-file (EOF) characters will terminate the local talk utility. Once the talk session has been terminated on one side, the other side of the talk session will be notified that the talk session has been terminated and will be able to do nothing except exit.
- Typing characters from LC_CTYPE classifications print or space will cause those characters to be sent to the recipient’s terminal.
- When and only when the stty iexten local mode is enabled, additional special control characters and multi-byte or single-byte characters are processed as printable characters if their wide character equivalents are printable.
- Typing other non-printable characters will cause them to be written to the recipient’s terminal as follows: control characters will appear as a caret (^) followed by the appropriate ASCII character, and characters with the high-order bit set will appear in “meta” notation. For example, ‘\003’ is displayed as ‘^C’ and ‘\372’ as ‘M–z’.

Permission to be a recipient of a talk message can be denied or granted by use of the mesg(1) utility. However, a user’s privilege may further constrain the domain of accessibility of other users’ terminals. Certain commands, such as pr(1), disallow messages in order to prevent interference with their output. talk will fail when the user lacks the appropriate privileges to perform the requested action.
Certain block-mode terminals do not have all the capabilities necessary to support the simultaneous exchange of messages required for talk. When this type of exchange cannot be supported on such terminals, the implementation may support an exchange with reduced levels of simultaneous interaction or it may report an error describing the terminal-related deficiency.

OPERANDS

The following operands are supported:

- **address**
  The recipient of the `talk` session. One form of `address` is the `username`, as returned by the `who(1)` utility. If you wish to talk to someone on your own machine, then `username` is just the person’s login name. If you wish to talk to a user on another host, then `username` is one of the following forms:

  - `host!user`
  - `host: user`
  - `user@host`

  although `user@host` is perhaps preferred.

- **terminal**
  If the recipient is logged in more than once, `terminal` can be used to indicate the appropriate terminal name. If `terminal` is not specified, the `talk` message will be displayed on one or more accessible terminals in use by the recipient. The format of `terminal` will be the same as that returned by `who`.

ENVIRONMENT VARIABLES

See `environ(5)` for descriptions of the following environment variables that affect the execution of `talk`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

- **TERM**
  Determine the name of the invoker’s terminal type. If this variable is unset or null, an unspecified terminal type will be used.

EXIT STATUS

The following exit values are returned:

- **0**
  Successful completion.

- **>0**
  An error occurred, or `talk` was invoked on a terminal incapable of supporting it.

FILES

- `/etc/hosts` host name database
- `/var/adm/utmpx` user and accounting information for `talk`

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWrcmds</td>
</tr>
</tbody>
</table>
Typing Control-L redraws the screen, while the erase, kill, and word kill characters will work in `talk` as normal. To exit, type an interrupt character; `talk` then moves the cursor to the bottom of the screen and restores the terminal to its previous state.
NAME
tar – create tape archives and add or extract files

SYNOPSIS
tar c [bBeEfFhiklnopPqvwX [0-7]] [block] [tarfile] [exclude-file]
   {-I include-file | -C directory | file | file...}
tar r [bBeEfFhiklnqvw@ [0-7]] [block] { -I include-file | -C directory | file |
   file...}  
tar t [BefFhiklnqvX [0-7]] [tarfile] [exclude-file] {-I include-file |
   file...}  
tar u [bBeEfFhiklnqvw@ [0-7]] [block] [tarfile] file...  
tar x [BefFhiklmnopqvwX [0-7]] [tarfile] [exclude-file] [file...]

DESCRIPTION
The tar command archives and extracts files to and from a single file called a tar file. A
 tarfile is usually a magnetic tape, but it can be any file. tar’s actions are controlled by
 the key argument. The key is a string of characters containing exactly one function
 letter (c, r, t, u, o r x) and zero or more function modifiers (letters or digits),
 depending on the function letter used. The key string contains no SPACE characters.
 Function modifier arguments are listed on the command line in the same order as their
 corresponding function modifiers appear in the key string.

The -I include-file, -C directory file, and file arguments specify which files or directories
 are to be archived or extracted. In all cases, appearance of a directory name refers to
 the files and (recursively) subdirectories of that directory. Arguments appearing within
 braces ({ }) indicate that one of the arguments must be specified.

OPTIONS
The following options are supported:

- I include-file
  Opens include-file containing a list of files, one per line, and treats it as if each file appeared separately on
  the command line. Be careful of trailing white spaces. Also beware of leading white spaces, since, for each line in
  the included file, the entire line (apart from the newline) will be used to match against the initial string
  of files to include. In the case where excluded files (see X function modifier) are also specified, they take
  precedence over all included files. If a file is specified in both the exclude-file and the include-file (or on the
  command line), it will be excluded.

- C directory file
  Performs a chdir (see cd(1)) operation on directory
  and performs the c (create) or r (replace) operation on
  file. Use short relative path names for file. If file is ‘.’,
  archive all files in directory. This option enables
  archiving files from multiple directories not related by
  a close common parent.

OPERANDS
The following operands are supported:

User Commands  1493
A path name of a regular file or directory to be archived (when the c, r or u functions are specified), extracted (x) or listed (t). When file is the path name of a directory, the action applies to all of the files and (recursively) subdirectories of that directory.

When a file is archived, and the E flag (see Function Modifiers) is not specified, the filename cannot exceed 256 characters. In addition, it must be possible to split the name between parent directory names so that the prefix is no longer than 155 characters and the name is no longer than 100 characters. If E is specified, a name of up to PATH_MAX characters may be specified.

For example, a file whose basename is longer than 100 characters could not be archived without using the E flag. A file whose directory portion is 200 characters and whose basename is 50 characters could be archived (without using E) if a slash appears in the directory name somewhere in character positions 151-156.

Function Letters

The function portion of the key is specified by one of the following letters:

- c  Create. Writing begins at the beginning of the tarfile, instead of at the end.
- r  Replace. The named files are written at the end of the tarfile. A file created with extended headers must be updated with extended headers (see E flag under Function Modifiers). A file created without extended headers cannot be modified with extended headers.
- t  Table of Contents. The names of the specified files are listed each time they occur in the tarfile. If no file argument is given, the names of all files and any associated extended attributes in the tarfile are listed. With the v function modifier, additional information for the specified files is displayed.
- u  Update. The named files are written at the end of the tarfile if they are not already in the tarfile, or if they have been modified since last written to that tarfile. An update can be rather slow. A tarfile created on a 5.x system cannot be updated on a 4.x system. A file created with extended headers must be updated with extended headers (see E flag under Function Modifiers). A file created without extended headers cannot be modified with extended headers.
- x  Extract or restore. The named files are extracted from the tarfile and written to the directory specified in the tarfile, relative to the current directory. Use the relative path names of files and directories to be extracted.

Absolute path names contained in the tar archive are unpacked using the absolute path names, that is, the leading forward slash (/) is not stripped off.
If a named file matches a directory whose contents has been written to the tarfile, this directory is recursively extracted. The owner, modification time, and mode are restored (if possible); otherwise, to restore owner, you must be the super-user. Character-special and block-special devices (created by *mknod*(1M)) can only be extracted by the super-user. If no file argument is given, the entire content of the tarfile is extracted. If the tarfile contains several files with the same name, each file is written to the appropriate directory, overwriting the previous one. Filename substitution wildcards cannot be used for extracting files from the archive. Rather, use a command of the form:

```
tar xvf ... /dev/rmt/0 `tar tf ... /dev/rmt/0 | \
grep 'pattern'`
```

When extracting tapes created with the *r* or *u* functions, directory modification times may not be set correctly. These same functions cannot be used with many tape drives due to tape drive limitations such as the absence of backspace or append capabilities.

When using the *x*, *u*, or *r* functions or the *X* function modifier, the named files must match exactly the corresponding files in the tarfile. For example, to extract */thisfile*, you must specify */thisfile*, and not *thisfile*. The *t* function displays how each file was archived.

### Function Modifiers

The characters below may be used in conjunction with the letter that selects the desired function.

- **b**: Blocking Factor. Use when reading or writing to raw magnetic archives (see *f* below). The block argument specifies the number of 512-byte tape blocks to be included in each read or write operation performed on the tarfile. The minimum is 1, the default is 20. The maximum value is a function of the amount of memory available and the blocking requirements of the specific tape device involved (see *mtio*(7I) for details.) The maximum cannot exceed \( \text{INT\_MAX}/512 (4194303) \).

  When a tape archive is being read, its actual blocking factor will be automatically detected, provided that it is less than or equal to the nominal blocking factor (the value of the block argument, or the default value if the b modifier is not specified). If the actual blocking factor is greater than the nominal blocking factor, a read error will result. See Example 5 in EXAMPLES.

- **B**: Block. Force *tar* to perform multiple reads (if necessary) to read exactly enough bytes to fill a block. This function modifier enables *tar* to work across the Ethernet, since pipes and sockets return partial blocks even when more data is coming. When reading from standard input, ‘-‘, this function modifier is selected by default to ensure that *tar* can recover from short reads.
Error. Exit immediately with a positive exit status if any unexpected errors occur. The SYSV3 environment variable overrides the default behavior. (See ENVIRONMENT VARIABLES section below.)

Write a tarfile with extended headers. (Used with c, r, or u options; ignored with t or x options.) When a tarfile is written with extended headers, the modification time is maintained with a granularity of microseconds rather than seconds. In addition, filenames no longer than PATH_MAX characters that could not be archived without E, and file sizes greater than 8GB, are supported. The E flag is required whenever the larger files and/or files with longer names, or whose UID/GID exceed 2097151, are to be archived, or if time granularity of microseconds is desired.

File. Use the tarfile argument as the name of the tarfile. If f is specified, /etc/default/tar is not searched. If f is omitted, tar will use the device indicated by the TAPE environment variable, if set; otherwise, it will use the default values defined in /etc/default/tar. The number matching the archiveN string is used as the output device with the blocking and size specifications from the file. For example,

tar -c 2/tmp/*

writes the output to the device specified as archive2 in /etc/default/tar.

If the name of the tarfile is ‘-’, tar writes to the standard output or reads from the standard input, whichever is appropriate. tar can be used as the head or tail of a pipeline. tar can also be used to move hierarchies with the command:

e xample% cd fromdir; tar cf .| (cd todir; tar xFb -)

With one F argument, tar excludes all directories named SCCS and RCS from the tarfile. With two arguments, FF, tar excludes all directories named SCCS and RCS, all files with .o as their suffix, and all files named errs, core, and a.out. The SYSV3 environment variable overrides the default behavior. (See ENVIRONMENT VARIABLES section below.)

Follow symbolic links as if they were normal files or directories. Normally, tar does not follow symbolic links.

Ignore directory checksum errors.

Requires tar to use the size argument as the size of an archive in kilobytes. This is useful when the archive is intended for a fixed size device such as floppy disks. Large files are then split across volumes if they do not fit in the specified size.
1 Link. Output error message if unable to resolve all links to the files being archived. If \l is not specified, no error messages are printed.

m Modify. The modification time of the file is the time of extraction. This function modifier is valid only with the x function.

n The file being read is a non-tape device. Reading of the archive is faster since tar can randomly seek around the archive.

o Ownership. Assign to extracted files the user and group identifiers of the user running the program, rather than those on tarfile. This is the default behavior for users other than root. If the o function modifier is not set and the user is root, the extracted files will take on the group and user identifiers of the files on tarfile (see chown(1) for more information). The o function modifier is only valid with the x function.

p Restore the named files to their original modes, and ACLs if applicable, ignoring the present umask(1). This is the default behavior if invoked as super-user with the x function letter specified. If super-user, SETUID, and sticky information are also extracted, and files are restored with their original owners and permissions, rather than owned by root. When this function modifier is used with the c function, ACLs are created in the tarfile along with other information. Errors will occur when a tarfile with ACLs is extracted by previous versions of tar.

P Suppress the addition of a trailing ”/” on directory entries in the archive.

q Stop after extracting the first occurrence of the named file. tar will normally continue reading the archive after finding an occurrence of a file.

v Verbose. Output the name of each file preceded by the function letter. With the t function, v provides additional information about the tarfile entries. The listing is similar to the format produced by the -l option of the ls(1) command.

w What. Output the action to be taken and the name of the file, then await the user’s confirmation. If the response is affirmative, the action is performed; otherwise, the action is not performed. This function modifier cannot be used with the t function.

X Exclude. Use the exclude-file argument as a file containing a list of relative path names for files (or directories) to be excluded from the tarfile when using the functions c, x, or t. Be careful of trailing white spaces. Also beware of leading white spaces, since, for each line in the excluded file, the entire line (apart from the newline) will be used to match against the initial string of files to exclude. Multiple X arguments may be used, with one exclude-file per
argument. In the case where included files (see -I include-file option) are also specified, the excluded files take precedence over all included files. If a file is specified in both the exclude-file and the include-file (or on the command line), it will be excluded.

@ Include extended attributes in archive. By default, tar does not place extended attributes in the archive. With this flag, tar will look for extended attributes on the files to be placed in the archive and add them to the archive. Extended attributes go in the archive as special files with a special type label. When this modifier is used with the x function, extended attributes are extracted from the tape along with the normal file data. Extended attribute files can only be extracted from an archive as part of a normal file extract. Attempts to explicitly extract attribute records are ignored.

[0-7] Select an alternative drive on which the tape is mounted. The default entries are specified in /etc/default/tar. If no digit or f function modifier is specified, the entry in /etc/default/tar with digit "0" is the default.

**USAGE**

See largefile(5) for the description of the behavior of tar when encountering files greater than or equal to 2 Gbyte ($2^{31}$ bytes).

The automatic determination of the actual blocking factor may be fooled when reading from a pipe or a socket (see the B function modifier below).

1/4" streaming tape has an inherent blocking factor of one 512-byte block. It can be read or written using any blocking factor.

This function modifier works for archives on disk files and block special devices, among others, but is intended principally for tape devices.

For information on tar header format, see archives(4).

**EXAMPLES**

**EXAMPLE 1** Creating an archive of your home directory

The following is an example using tar to create an archive of your home directory on a tape mounted on drive /dev/rmt/0:

```
example% cd
example% tar cvf /dev/rmt/0 .
```

The c function letter means create the archive. The v function modifier outputs messages explaining what tar is doing. The f function modifier indicates that the tarfile is being specified (/dev/rmt/0 in this example). The dot (.) at the end of the command line indicates the current directory and is the argument of the f function modifier.

Display the table of contents of the tarfile with the following command:
EXAMPLE 1 Creating an archive of your home directory (Continued)

```bash
example% tar tvf /dev/rmt/0
```

The output will be similar to the following for the POSIX locale:

```
rw−r−−r−− 1677/40 2123 Nov 7 18:15 1985 ./test.c ...
```

The columns have the following meanings:

- Column 1 is the access permissions to ./test.c
- Column 2 is the user-id/group-id of ./test.c
- Column 3 is the size of ./test.c in bytes
- Column 4 is the modification date of ./test.c. When the LC_TIME category is not set to the POSIX locale, a different format and date order field may be used.
- Column 5 is the name of ./test.c

To extract files from the archive:

```bash
example% tar xvf /dev/rmt/0
```

If there are multiple archive files on a tape, each is separated from the following one by an EOF marker. To have tar read the first and second archives from a tape with multiple archives on it, the non-rewinding version of the tape device name must be used with the $f$ function modifier, as follows:

```bash
example% tar xvfp /dev/rmt/0n read first archive from tape
example% tar xvfp /dev/rmt/0n read second archive from tape
```

Notice that in some earlier releases, the above scenario did not work correctly, and intervention with mt(1) between tar invocations was necessary. To emulate the old behavior, use the non-rewind device name containing the letter b for BSD behavior. See the Close Operations section of the mtio(7I) manual page.

EXAMPLE 2 Archiving files from /usr/include and from /etc to default tape drive 0

To archive files from /usr/include and from /etc to default tape drive 0:

```bash
example% tar c -C/usr include -C/etc
```

The table of contents from the resulting tarfile would produce output like the following:
EXAMPLE 2 Archiving files from /usr/include and from /etc to default tape drive 0 (Continued)

```
include/
include/a.out.h
and all the other files in /usr/include ...
./chown and all the other files in /etc
```

To extract all files in the include directory:

```
example% tar xv include
x include/, 0 bytes, 0 tape blocks \
    and all files under include ...
```

EXAMPLE 3 Transferring files across the network

The following is an example using tar to transfer files across the network. First, here is how to archive files from the local machine (example) to a tape on a remote system (host):

```
example% tar cvfb -20 files | \rsh host dd of=/dev/rmt/0 obs=20b
messages from tar
example%
```

In the example above, we are creating a tarfile with the c key letter, asking for verbose output from tar with the v function modifier, specifying the name of the output tarfile using the f function modifier (the standard output is where the tarfile appears, as indicated by the ‘−’ sign), and specifying the blocksize (20) with the b function modifier. If you want to change the blocksize, you must change the blocksize arguments both on the tar command and on the dd command.

EXAMPLE 4 Retrieving files from a tape on the remote system back to the local system

The following is an example that uses tar to retrieve files from a tape on the remote system back to the local system:

```
exmaple% rsh -n host dd if=/dev/rmt/0 bs=20b | \tar xvBfb -20 files
messages from tar
example%
```

In the example above, we are extracting from the tarfile with the x key letter, asking for verbose output from tar with the v function modifier, telling tar it is reading from a pipe with the b function modifier, specifying the name of the input tarfile using the f function modifier (the standard input is where the tarfile appears, as indicated by the ‘−’ sign), and specifying the blocksize (20) with the b function modifier.
EXAMPLE 5 Creating an archive of the home directory

The following example creates an archive of the home directory on /dev/rmt/0 with an actual blocking factor of 19:

example% tar cvfb /dev/rmt/0 19 $HOME

To recognize this archive’s actual blocking factor without using the b function modifier:

example% tar tvf /dev/rmt/0
.tar: blocksize = 19
...

To recognize this archive’s actual blocking factor using a larger nominal blocking factor:

example% tar tvf /dev/rmt/0 30
.tar: blocksize = 19
...

Attempt to recognize this archive’s actual blocking factor using a nominal blocking factor that is too small:

example% tar tvf /dev/rmt/0 10
.tar: tape read error

ENVIRONMENT VARIABLES

SYSV3 This variable is used to override the default behavior of tar, provide compatibility with INTERACTIVE UNIX Systems and SCO UNIX installation scripts, and should not be used in new scripts. (It is intended for compatibility purposes only.) When set, the following options behave differently:

- F filename Uses filename to obtain a list of command line switches and files on which to operate.

- e Prevents files from being split across volumes. If there is insufficient room on one volume, tar prompts for a new volume. If the file will not fix on the new volume, tar exits with an error.

See environ(5) for descriptions of the following environment variables that affect the execution of tar: LC_CTYPE, LC_MESSAGES, LC_TIME, TZ, and NLSPATH.

EXIT STATUS The following exit values are returned:

0 Successful completion.

>0 An error occurred.
Settings may look like this:

```
archive0=/dev/rmt/0
archive1=/dev/rmt/0n
archive2=/dev/rmt/1
archive3=/dev/rmt/1n
archive4=/dev/rmt/0
archive5=/dev/rmt/0n
archive6=/dev/rmt/1
archive7=/dev/rmt/1n
```

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Stable</td>
</tr>
</tbody>
</table>

SEE ALSO

ar(1), basename(1), cd(1), chown(1), cpio(1), csh(1), dirname(1), ls(1), mt(1),
pax(1), setfacl(1), umask(1), mknod(1M), vold(1M), archives(4),
attributes(5), environ(5), fsattr(5), largefile(5), mtio(7I)

DIAGNOSTICS

Diagnostic messages are output for bad key characters and tape read/write errors, and
for insufficient memory to hold the link tables.

NOTES

There is no way to access the n-th occurrence of a file.

Tape errors are handled ungracefully.

When the Volume Management daemon is running, accesses to floppy devices
through the conventional device names (for example, /dev/rdiskette) may not
succeed. See vold(1M) for further details.
The *tar* archive format allows UIDs and GIDs up to 2097151 to be stored in the archive header. Files with UIDs and GIDs greater than this value will be archived with the UID and GID of 60001.

If an archive is created that contains files whose names were created by processes running in multiple locales, a single locale that uses a full 8-bit codeset (for example, the *en_US* locale) should be used both to create the archive and to extract files from the archive.

Neither the `-r` option nor the `-u` option can be used with quarter-inch archive tapes, since these tape drives cannot backspace.
NAME  tbl – format tables for nroff or troff

SYNOPSIS  tbl [-me] [-mm] [-ms] [filename]...

DESCRIPTION  tbl is a preprocessor for formatting tables for nroff(1) or troff(1). The input filenames are copied to the standard output, except that lines between .TS and .TE command lines are assumed to describe tables and are reformatted.

If no arguments are given, tbl reads the standard input, so tbl may be used as a filter. When tbl is used with eqn(1) or neqn, the tbl command should be first, to minimize the volume of data passed through pipes.

OPTIONS  
- me  Copy the -me macro package to the front of the output file.
- mm  Copy the -mm macro package to the front of the output file.
- ms  Copy the -ms macro package to the front of the output file.

EXAMPLES  EXAMPLE 1 Using tbl

As an example, letting ‘@’ (at-sign) represent a TAB, which should be typed as an actual TAB character in the input file

```
.TS
 c s s
 c c c
 1 n n.
Household Population
 Town # Households
 @# Numbers @ Size
Bedminster 789@3.26
Bernards Twp. 3087@3.74
Bernardsville 2018@3.30
Bound Brook 3425@3.04
Branchburg 1644@3.49
.TE
```
yields

```

Household Population

<table>
<thead>
<tr>
<th>Town</th>
<th>Households</th>
<th>Number</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bedminster</td>
<td></td>
<td>789</td>
<td>3.26</td>
</tr>
<tr>
<td>Bernards Twp.</td>
<td></td>
<td>3087</td>
<td>3.74</td>
</tr>
<tr>
<td>Bernardsville</td>
<td></td>
<td>2018</td>
<td>3.30</td>
</tr>
<tr>
<td>Bound Brook</td>
<td></td>
<td>3425</td>
<td>3.04</td>
</tr>
<tr>
<td>Branchburg</td>
<td></td>
<td>1644</td>
<td>3.49</td>
</tr>
</tbody>
</table>

```
FILES

/usr/share/lib/tmac/e -me macros
/usr/share/lib/tmac/m -mm macros
/usr/share/lib/tmac/s -ms macros

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWdoc</td>
</tr>
</tbody>
</table>

SEE ALSO  eqn(1), nroff(1), troff(1), attributes(5)
tcopy(1)

NAME       tcopy – copy a magnetic tape
SYNOPSIS   tcopy source [destination]
DESCRIPTION The tcopy utility copies the magnetic tape mounted on the tape drive specified by the source argument. The only assumption made about the contents of a tape is that there are two tape marks at the end.

When only a source drive is specified, tcopy scans the tape, and displays information about the sizes of records and tape files. If a destination is specified, tcopy makes a copies the source tape onto the destination tape, with blocking preserved. As it copies, tcopy produces the same output as it does when only scanning a tape.

The tcopy utility requires the use of Berkeley-compatible device names. For example,

eexample$ tcopy /dev/rmt/1b /dev/rmt/2b

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
</tbody>
</table>

SEE ALSO       mt(1), ioctl(2), attributes(5)

NOTES          tcopy will only run on systems supporting an associated set of ioctl(2) requests.
NAME  tee – replicate the standard output

SYNOPSIS  tee [-ai] [file...]

DESCRIPTION  The tee utility will copy standard input to standard output, making a copy in zero or more files. tee will not buffer its output. The options determine if the specified files are overwritten or appended to.

OPTIONS  The following options are supported.
  -a  Append the output to the files rather than overwriting them.
  -i  Ignore interrupts.

OPERANDS  The following operands are supported:
  file  A path name of an output file. Processing of at least 13 file operands will be supported.

USAGE  See largefile(5) for the description of the behavior of tee when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

ENVIRONMENT VARIABLES  See environ(5) for descriptions of the following environment variables that affect the execution of tee: LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS  The following exit values are returned:
  0  The standard input was successfully copied to all output files.
  >0  The number of files that could not be opened or whose status could not be obtained.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

SEE ALSO  cat(1), attributes(5), environ(5), largefile(5),
The `telnet` utility communicates with another host using the TELNET protocol. If `telnet` is invoked without arguments, it enters command mode, indicated by its prompt, `telnet>`. In this mode, it accepts and executes its associated commands. See USAGE, `telnet` Commands, below. If it is invoked with arguments, it performs an open command with those arguments.

If, for example, a `host` is specified as `@hop1@hop2@host`, the connection goes through hosts `hop1` and `hop2`, using loose source routing to end at `host`. If a leading `!` is used, the connection follows strict source routing. Note that when `telnet` uses IPv6, it can only use loose source routing, and the connection ignores the `!`.

Once a connection has been opened, `telnet` enters input mode. In this mode, text typed is sent to the remote host. The input mode entered will be either "line mode", "character at a time", or "old line by line", depending upon what the remote system supports.

In "line mode", character processing is done on the local system, under the control of the remote system. When input editing or character echoing is to be disabled, the remote system will relay that information. The remote system will also relay changes to any special characters that happen on the remote system, so that they can take effect on the local system.

In "character at a time" mode, most text typed is immediately sent to the remote host for processing.

In "old line by line" mode, all text is echoed locally, and (normally) only completed lines are sent to the remote host. The "local echo character" (initially `^E`) may be used to turn off and on the local echo. (Use this mostly to enter passwords without the password being echoed.).

If the "line mode" option is enabled, or if the `localchars` toggle is TRUE (the default in "old line by line" mode), the user's `quit`, `intr`, and `flush` characters are trapped locally, and sent as TELNET protocol sequences to the remote side. If "line mode" has ever been enabled, then the user's `suspend` and `eof` are also sent as TELNET protocol sequences. `quit` is then sent as a TELNET ABORT instead of BREAK. The options `toggle autoflush` and `toggle autosynch` cause this action to flush subsequent output to the terminal (until the remote host acknowledges the TELNET sequence); and to flush previous terminal input, in the case of `quit` and `intr`.

While connected to a remote host, the user can enter `telnet` command mode by typing the `telnet` escape character (initially `^]`). When in command mode, the normal terminal editing conventions are available. Pressing RETURN at the `telnet` command prompt causes `telnet` to exit command mode.

### OPTIONS

The following options are supported:
-8  Specifies an 8-bit data path. Negotiating the TELNET BINARY option is attempted for both input and output.

-c  Disables the reading of the user's telnetrc file. (See the toggle skiprc command on this reference page.)

-d  Sets the initial value of the debug toggle to TRUE.

-e escape_char  Sets the initial escape character to escape_char. escape_char may also be a two character sequence consisting of '^
followed by one character. If the second character is '?', the DEL character is selected. Otherwise, the second character is converted to a control character and used as the escape character. If the escape character is the null string (that is, -e’”), it is disabled.

-E  Stops any character from being recognized as an escape character.

-l user  When connecting to a remote system that understands the ENVIRON option, then user will be sent to the remote system as the value for the ENVIRON variable USER.

-L  Specifies an 8-bit data path on output. This causes the BINARY option to be negotiated on output.

-n tracefile  Opens tracefile for recording trace information. See the set tracefile command below.

-r  Specifies a user interface similar to rlogin. In this mode, the escape character is set to the tilde (~) character, unless modified by the -e option. The rlogin escape character is only recognized when it is preceded by a carriage return. In this mode, the telnet escape character, normally ‘^]’, must still precede a telnet command. The rlogin escape character can also be followed by ‘.’ or ‘^Z’, and, like rlogin(1), closes or suspends the connection, respectively. This option is an uncommitted interface and may change in the future.

The commands described in this section are available with telnet. It is necessary to type only enough of each command to uniquely identify it. (This is also true for arguments to the mode, set, toggle, unset, environ, and display commands.)

open [ -l user ] [ [!] @hop1 [@hop2 ...]@host [ port ]

Open a connection to the named host. If no port number is specified, telnet will attempt to contact a TELNET server at the default port. The host specification may be either a host name (see hosts(4), ipnodes(4)) or an Internet address specified in the "dot notation" (see inet(7P) or inet6(7P)). If the host is specified as @hop1@hop2@host, the connection goes through hosts hop1 and hop2, using loose source routing to end at host. The ‘’ symbol is required as a separator between the hosts specified. If a leading ! is used with IPv4, the connection follows strict source routing.
The `-l` option passes the `user` as the value of the `ENVIRON` variable `USER` to the remote system.

`close`
Close any open TELNET session and exit `telnet`. An EOF (in command mode) will also close a session and exit.

`quit`
Same as `close`.

`z`
Suspend `telnet`. This command only works when the user is using a shell that supports job control, such as `sh(1)`.

`mode type`
The remote host is asked for permission to go into the requested mode. If the remote host is capable of entering that mode, the requested mode will be entered. The argument `type` is one of the following:

- `character`
  Disable the TELNET LINEMODE option, or, if the remote side does not understand the LINEMODE option, then enter "character at a time" mode.

- `line`
  Enable the TELNET LINEMODE option, or, if the remote side does not understand the LINEMODE option, then attempt to enter "old-line-by-line" mode.

- `isig (-isig)`
  Attempt to enable (disable) the TRAPSIG mode of the LINEMODE option. This requires that the LINEMODE option be enabled.

- `edit (-edit)`
  Attempt to enable (disable) the EDIT mode of the LINEMODE option. This requires that the LINEMODE option be enabled.

- `softtabs (-softtabs)`
  Attempt to enable (disable) the SOFT_TAB mode of the LINEMODE option. This requires that the LINEMODE option be enabled.

- `litecho (-litecho)`
  Attempt to enable (disable) the LIT_ECHO mode of the LINEMODE option. This requires that the LINEMODE option be enabled.

- `?`
  Prints out help information for the `mode` command.

`status`
Show the current status of `telnet`. This includes the peer one is connected to, as well as the current mode.

`display`
`[ argument ... ]` Display all, or some, of the set and toggle values (see `toggle argument ...`).
Get help. With no arguments, `telnet` prints a help summary. If a command is specified, `telnet` will print the help information for just that command.

```
[command] Get help. With no arguments, telnet prints a help summary. If a command is specified, telnet will print the help information for just that command.
```

Send one or more special character sequences to the remote host. The following are the arguments that can be specified (more than one argument may be specified at a time):

- `escape` Send the current `telnet` escape character (initially `^]`).
- `synch` Send the `TELNET SYNCH` sequence. This sequence discards all previously typed, but not yet read, input on the remote system. This sequence is sent as TCP urgent data and may not work if the remote system is a 4.2 BSD system. If it does not work, a lower case ‘r’ may be echoed on the terminal.
- `brk` or `break` Send the `TELNET BRK (Break)` sequence, which may have significance to the remote system.
- `ip` Send the `TELNET IP (Interrupt Process)` sequence, which aborts the currently running process on the remote system.
- `abort` Send the `TELNET ABORT (Abort Process)` sequence.
- `ao` Send the `TELNET AO (Abort Output)` sequence, which flushes all output from the remote system to the user’s terminal.
- `ayt` Send the `TELNET AYT (Are You There)` sequence, to which the remote system may or may not respond.
- `ec` Send the `TELNET EC (Erase Character)` sequence, which erases the last character entered.
- `el` Send the `TELNET EL (Erase Line)` sequence, which should cause the remote system to erase the line currently being entered.
- `eof` Send the `TELNET EOF (End Of File)` sequence.
- `eot` Send the `TELNET EOR (End Of Record)` sequence.
- `ga` Send the `TELNET GA (Go Ahead)` sequence, which probably has no significance for the remote system.
- `getstatus` If the remote side supports the `TELNET STATUS` command, `getstatus` will send the subnegotiation to request that the server send its current option status.
- `nop` Send the `TELNET NOP (No Operation)` sequence.
- `susp` Send the `TELNET SUSP (Suspend Process)` sequence.
do option
dont option
will option
wont option

Send the TELNET protocol option negotiation indicated. Option may be the text name of the protocol option, or the number corresponding to the option. The command will be silently ignored if the option negotiation indicated is not valid in the current state. If the option is given as 'help' or '?', the list of option names known is listed. This command is mostly useful for unusual debugging situations.

? 
Print out help information for the send command.

set argument [ value ]
unset argument

Set any one of a number of telnet variables to a specific value. The special value "off" turns off the function associated with the variable. The values of variables may be interrogated with the display command. If value is omitted, the value is taken to be true, or "on". If the unset form is used, the value is taken to be false, or "off."

The variables that may be specified are:

echo This is the value (initially ^E) that, when in "line by line" mode, toggles between local echoing of entered characters for normal processing, and suppressing echoing of entered characters, for example, entering a password.

escape This is the telnet escape character (initially ^I) that enters telnet command mode when connected to a remote system.

interrupt If telnet is in localchars mode (see toggle, localchars) and the interrupt character is typed, a TELNET IP sequence (see send and ip) is sent to the remote host. The initial value for the interrupt character is taken to be the terminal's intr character.

quit If telnet is in localchars mode and the quit character is typed, a TELNET BRK sequence (see send, brk) is sent to the remote host. The initial value for the quit character is taken to be the terminal's quit character.

flushoutput If telnet is in localchars mode and the flushoutput character is typed, a TELNET AO sequence (see send, ao) is sent to the remote host. The initial value for the flush character is taken to be the terminal's flush character.

erase If telnet is in localchars mode and operating in "character at a time" mode, then when the erase character is typed, a TELNET EC sequence (see send, ec) is sent to the remote system. The initial value for the erase character is taken to be the terminal's erase character.
kill

If telnet is in localchars mode and operating in "character at a time" mode, then when the kill character is typed, a TELNET EL sequence (see send, el) is sent to the remote system. The initial value for the kill character is taken to be the terminal's kill character.

eof

If telnet is operating in "line by line" mode, entering the eof character as the first character on a line sends this character to the remote system. The initial value of eof is taken to be the terminal's eof character.

ayt

If telnet is in localchars mode, or LINEMODE is enabled, and the status character is typed, a TELNET AYT ("Are You There") sequence is sent to the remote host. (See send, ayt above.) The initial value for ayt is the terminal's status character.

forw1
forw2

If telnet is operating in LINEMODE, and the forw1 or forw2 characters are typed, this causes the forwarding of partial lines to the remote system. The initial values for the forwarding characters come from the terminal's eol and eol2 characters.

lnext

If telnet is operating in LINEMODE or "old line by line" mode, then the lnext character is assumed to be the terminal's lnext character. The initial value for the lnext character is taken to be the terminal's lnext character.

reprint

If telnet is operating in LINEMODE or "old line by line" mode, then the reprint character is assumed to be the terminal's reprint character. The initial value for reprint is taken to be the terminal's reprint character.

rlogin

This is the rlogin escape character. If set, the normal telnet escape character is ignored, unless it is preceded by this character at the beginning of a line. The rlogin character, at the beginning of a line followed by a "." closes the connection. When followed by a "^Z", the rlogin command suspends the telnet command. The initial state is to disable the rlogin escape character.

start

If the TELNET TOGGLE-FLOW-CONTROL option has been enabled, then the start character is taken to be the terminal's start character. The initial value for the kill character is taken to be the terminal's start character.

stop

If the TELNET TOGGLE-FLOW-CONTROL option has been enabled, then the stop character is taken to be the terminal's stop character. The initial value for the kill character is taken to be the terminal's stop character.
If telnet is in localchars mode, or LINEMODE is enabled, and the suspend character is typed, a TELNET SUSP sequence (see send, susp above) is sent to the remote host. The initial value for the suspend character is taken to be the terminal’s suspend character.

This is the file to which the output, generated when the netdata or the debug option is TRUE, will be written. If tracefile is set to ” ”, then tracing information will be written to standard output (the default).

If telnet is operating in LINEMODE or "old line by line" mode, then this character is taken to be the terminal’s worderase character. The initial value for the worderase character is taken to be the terminal’s worderase character.

Displays the legal set and unset commands.

The slc (Set Local Characters) command is used to set or change the state of special characters when the TELNET LINEMODE option has been enabled. Special characters are characters that get mapped to TELNET commands sequences (like ip or quit) or line editing characters (like erase and kill). By default, the local special characters are exported. The following values for state are valid:

Verifies the settings for the current special characters. The remote side is requested to send all the current special character settings. If there are any discrepancies with the local side, the local settings will switch to the remote values.

Switches to the local defaults for the special characters. The local default characters are those of the local terminal at the time when telnet was started.

Switches to the remote defaults for the special characters. The remote default characters are those of the remote system at the time when the TELNET connection was established.

Prints out help information for the slc command.

...Toggle between TRUE and FALSE the various flags that control how telnet responds to events. More than one argument may be specified. The state of these flags may be interrogated with the display command. Valid arguments are:

If autoflush and localchars are both TRUE, then when the ao, intr, or quit characters are recognized (and transformed into TELNET sequences; see set for details), telnet refuses to display any data on the user’s terminal until the remote system acknowledges (using a TELNET Timing Mark option) that it has processed those TELNET sequences. The initial value for this
toggle is TRUE if the terminal user has not done an "stty noflsh". Otherwise, the value is FALSE (see stty(1)).

**autosynch**
If autosynch and localchars are both TRUE, then when either the interrupt or quit characters are typed (see set for descriptions of interrupt and quit), the resulting TELNET sequence sent is followed by the TELNET SYNCH sequence. This procedure should cause the remote system to begin throwing away all previously typed input until both of the TELNET sequences have been read and acted upon. The initial value of this toggle is FALSE.

**binary**
Enable or disable the TELNET BINARY option on both input and output.

**inbinary**
Enable or disable the TELNET BINARY option on input.

**outbinary**
Enable or disable the TELNET BINARY option on output.

**crlf**
Determines how carriage returns are sent. If the value is TRUE, then carriage returns will be sent as \(<CR><LF>\). If the value is FALSE, then carriage returns will be sent as \(<CR><NUL>\). The initial value for this toggle is FALSE.

**crmod**
Toggle RETURN mode. When this mode is enabled, most RETURN characters received from the remote host will be mapped into a RETURN followed by a line feed. This mode does not affect those characters typed by the user, only those received from the remote host. This mode is useful only for remote hosts that send RETURN but never send LINEFEED. The initial value for this toggle is FALSE.

**debug**
Toggle socket level debugging (only available to the super-user). The initial value for this toggle is FALSE.

**localchars**
If this toggle is TRUE, then the flush, interrupt, quit, erase, and kill characters (see set) are recognized locally, and transformed into appropriate TELNET control sequences, respectively ao, ip, brk, ec, and el (see send). The initial value for this toggle is TRUE in "line by line" mode, and FALSE in "character at a time" mode. When the LINEMODE option is enabled, the value of localchars is ignored, and assumed always to be TRUE. If LINEMODE has ever been enabled, then quit is sent as abort, and eof and suspend are sent as eof and susp (see send above).

**netdata**
Toggle the display of all network data (in hexadecimal format). The initial value for this toggle is FALSE.

**options**
Toggle the display of some internal TELNET protocol processing (having to do with telnet options). The initial value for this toggle is FALSE.
prettydump When the netdata toggle is enabled, if prettydump is enabled, the output from the netdata command will be formatted in a more user readable format. Spaces are put between each character in the output. The beginning of any TELNET escape sequence is preceded by an asterisk (*) to aid in locating them.

skiprc When the skiprc toggle is TRUE, TELNET skips the reading of the .telnetrc file in the user’s home directory when connections are opened. The initial value for this toggle is FALSE.

termdata Toggles the display of all terminal data (in hexadecimal format). The initial value for this toggle is FALSE.

? Display the legal toggle commands.

environ argument...

The environ command is used to manipulate variables that may be sent through the TELNET ENVIRON option. The initial set of variables is taken from the users environment. Only the DISPLAY and PRINTER variables are exported by default. Valid arguments for the environ command are:

define variable value Define variable to have a value of value. Any variables defined by this command are automatically exported. The value may be enclosed in single or double quotes, so that tabs and spaces may be included.

undefine variable Remove variable from the list of environment variables.

export variable Mark the variable to be exported to the remote side.

unexport variable Mark the variable to not be exported unless explicitly requested by the remote side.

list List the current set of environment variables. Those marked with an asterisk (*) will be sent automatically. Other variables will be sent only if explicitly requested.

? Prints out help information for the environ command.

logout Sends the telnet logout option to the remote side. This command is similar to a close command. However, if the remote side does not support the logout option, nothing happens. If, however, the remote side does support the logout option, this command should cause the remote side to close the TELNET connection. If the remote side also supports the concept of suspending a user’s session for later reattachment, the logout argument indicates that the remote side should
terminate the session immediately.

**FILES**

`$HOME/.telnetrc`

file that contains commands to be executed before initiating a telnet session

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWtnetc</td>
</tr>
</tbody>
</table>

**SEE ALSO**

`rlogin(1), sh(1), stty(1), hosts(4), ipnodes(4), nologin(4), telnetrc(4), attributes(5), inet(7P), inet6(7P)`

**DIAGNOSTICS**

`NO LOGINS: System going down in N minutes`

The machine is in the process of being shut down and logins have been disabled.

**NOTES**

On some remote systems, echo has to be turned off manually when in "line by line" mode.

In "old line by line" mode, or LINEMODE, the terminal’s EOF character is only recognized (and sent to the remote system) when it is the first character on a line.
The `test` utility evaluates the `condition` and indicates the result of the evaluation by its exit status. An exit status of zero indicates that the condition evaluated as true and an exit status of 1 indicates that the condition evaluated as false.

In the first form of the utility shown using the SYNOPSIS:

```
test [ condition ]
```

the square brackets denote that `condition` is an optional operand and are not to be entered on the command line.

In the second form of the utility shown using the SYNOPSIS:

```
[ [ condition ] ]
```

the first open square bracket, `[`, is the required utility name. `condition` is optional, as denoted by the inner pair of square brackets. The final close square bracket, `]`, is a required operand.

See `largefile(5)` for the description of the behavior of `test` when encountering files greater than or equal to 2 Gbyte (2³¹ bytes).

The `test` and `[` utilities evaluate the condition `condition` and, if its value is true, set exit status to 0. Otherwise, a non-zero (false) exit status is set. `test` and `[` also set a non-zero exit status if there are no arguments. When permissions are tested, the effective user ID of the process is used.

All operators, flags, and brackets (brackets used as shown in the last SYNOPSIS line) must be separate arguments to these commands. Normally these arguments are separated by spaces.

The primaries listed below with two elements of the form:

```
-primary_operator primary_operand
```
are known as **unary primaries**. The primaries with three elements in either of the two forms:

```
primary_operand primary_operator primary_operand
primary_operand primary_operator primary_operand
```

are known as **binary primaries**.

If any file operands except for `-h` and `-L` primaries refer to symbolic links, the symbolic link is expanded and the test is performed on the resulting file.

If you test a file you own (the `-r` `-w` or `-x` tests), but the permission tested does not have the *owner* bit set, a non-zero (false) exit status will be returned even though the file may have the *group* or *other* bit set for that permission.

The `=` and `!=` primaries have a higher precedence than the unary primaries. The `=` and `!=` primaries always expect arguments; therefore, `=` and `!=` cannot be used as an argument to the unary primaries.

The following primaries can be used to construct condition:

- `-a file` True, if *file* exists. (Not available in *sh*).
- `-b file` True if *file* exists and is a block special file.
- `-c file` True if *file* exists and is a character special file.
- `-d file` True if *file* exists and is a directory.
- `-e file` True if *file* exists.
- `-f file` True if *file* exists and is a regular file. Alternatively, if `/usr/bin/sh` users specify `/usr/ucb` before `/usr/bin` in their `PATH` environment variable, then `test` will return true if *file* exists and is (not-a-directory). The `csh test` and `{ built-ins` always use this alternative behavior.
- `-g file` True if *file* exists and its set group ID flag is set.
- `-G file` True if *file* exists and its group matches the effective group ID of this process. (Not available in *sh*.)
- `-h file` True if *file* exists and is a symbolic link.
- `-k file` True if *file* exists and has its sticky bit set.
- `-L file` True, if *file* exists and is a symbolic link.
- `-n string` True if the length of *string* is non-zero.
- `-o option` True if option named *option* is on. (Not available in *csh* or *sh*.)
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-o file</td>
<td>True if file exists and is owned by the effective user ID of this process. (Not available in $sh.)</td>
</tr>
<tr>
<td>-p file</td>
<td>True if file is a named pipe (FIFO).</td>
</tr>
<tr>
<td>-x file</td>
<td>True if file exists and is readable.</td>
</tr>
<tr>
<td>-a file</td>
<td>True if file exists and has a size greater than zero.</td>
</tr>
<tr>
<td>-S file</td>
<td>True if file exists and is a socket. (Not available in $sh.)</td>
</tr>
<tr>
<td>-t [file_descriptor]</td>
<td>True if the file whose file descriptor number is file_descriptor is open and is associated with a terminal. If file_descriptor is not specified, 1 is used as a default value.</td>
</tr>
<tr>
<td>-u file</td>
<td>True if file exists and its set-user-ID flag is set.</td>
</tr>
<tr>
<td>-w file</td>
<td>True if file exists and is writable. True will indicate only that the write flag is on. The file will not be writable on a read-only file system even if this test indicates true.</td>
</tr>
<tr>
<td>-x file</td>
<td>True if file exists and is executable. True will indicate only that the execute flag is on. If file is a directory, true indicates that file can be searched.</td>
</tr>
<tr>
<td>-z string</td>
<td>True if the length of string string is zero.</td>
</tr>
<tr>
<td>file1 -nt file2</td>
<td>True, if file1 exists and is newer than file2. (Not available in $sh.)</td>
</tr>
<tr>
<td>file1 -ot file2</td>
<td>True, if file1 exists and is older than file2. (Not available in $sh.)</td>
</tr>
<tr>
<td>file1 -ef file2</td>
<td>True, if file1 and file2 exist and refer to the same file. (Not available in $sh.)</td>
</tr>
<tr>
<td>string</td>
<td>True if the string string is not the null string.</td>
</tr>
<tr>
<td>string1 = string2</td>
<td>True if the strings string1 and string2 are identical.</td>
</tr>
<tr>
<td>string1 != string2</td>
<td>True if the strings string1 and string2 are not identical.</td>
</tr>
<tr>
<td>n1 -eq n2</td>
<td>True if the integers n1 and n2 are algebraically equal.</td>
</tr>
<tr>
<td>n1 -ne n2</td>
<td>True if the integers n1 and n2 are not algebraically equal.</td>
</tr>
<tr>
<td>n1 -gt n2</td>
<td>True if the integer n1 is algebraically greater than the integer n2.</td>
</tr>
<tr>
<td>n1 -ge n2</td>
<td>True if the integer n1 is algebraically greater than or equal to the integer n2.</td>
</tr>
<tr>
<td>n1 -lt n2</td>
<td>True if the integer n1 is algebraically less than the integer n2.</td>
</tr>
</tbody>
</table>
test

-\le \ n2 \quad \text{True if the integer } n1 \text{ is algebraically less than or equal to the integer } n2.

\text{condition1 } \mathord{-} a \text{ condition2} \quad \text{True if both condition1 and condition2 are true. The } -a \text{ binary primary is left associative and has higher precedence than the } -o \text{ binary primary.}

\text{condition1 } \mathord{-} o \text{ condition2} \quad \text{True if either condition1 or condition2 is true. The } -o \text{ binary primary is left associative.}

These primaries can be combined with the following operators:

\! \text{condition} \quad \text{True if condition is false.}

( \text{condition} ) \quad \text{True if condition is true. The parentheses ( ) can be used to alter the normal precedence and associativity. Notice also that parentheses are meaningful to the shell and, therefore, must be quoted.}

The algorithm for determining the precedence of the operators and the return value that will be generated is based on the number of arguments presented to test. (However, when using the [ . . . ] form, the right-bracket final argument will not be counted in this algorithm.)

In the following list, $1$, $2$, $3$, and $4$ represent the arguments presented to test as a condition, condition1, or condition2.

0 arguments: Exit false (1).

1 argument: Exit true (0) if $1$ is not null; otherwise, exit false.

2 arguments:
- If $1$ is !, exit true if $2$ is null, false if $2$ is not null.
- If $1$ is a unary primary, exit true if the unary test is true, false if the unary test is false.
- Otherwise, produce unspecified results.

3 arguments:
- If $2$ is a binary primary, perform the binary test of $1$ and $3$.
- If $1$ is !, negate the two-argument test of $2$ and $3$.
- Otherwise, produce unspecified results.

4 arguments:
- If $1$ is !, negate the three-argument test of $2$, $3$, and $4$.
- Otherwise, the results are unspecified.

Usage

Scripts should be careful when dealing with user-supplied input that could be confused with primaries and operators. Unless the application writer knows all the cases that produce input to the script, invocations like test "$1" -a "$2" should be written as test "$1" && test "$2" to avoid problems if a user supplied values such as $1$ set to ! and $2$ set to the null string. That is, in cases where maximal
Portability is of concern, replace `test expr1 -a expr2` with `test expr1 && test expr2`, and replace `test expr1 -o expr2` with `test expr1 || test expr2`. But notice that, in `test`, `-a` has higher precedence than `-o`, while `&&` and `||` have equal precedence in the shell.

Parentheses or braces can be used in the shell command language to effect grouping.

Parentheses must be escaped when using `sh`. For example:

```
   test \( expr1 -a expr2 \) -o expr3
```

This command is not always portable outside XSI-conformant systems. The following form can be used instead:

```
   ( test expr1 && test expr2 ) || test expr3
```

The two commands:

```
   test "$1"
   test ! "$1"
```

could not be used reliably on some historical systems. Unexpected results would occur if such a string condition were used and `$1` expanded to `!`, `(`, or a known unary primary. Better constructs are, respectively,

```
   test -n "$1"
   test -z "$1"
```

Historical systems have also been unreliable given the common construct:

```
   test "$response" = "expected string"
```

One of the following is a more reliable form:

```
   test "X$response" = "Xexpected string"
   test "expected string" = "$response"
```

Notice that the second form assumes that `expected string` could not be confused with any unary primary. If `expected string` starts with `-`, `(`, `!` or even `=`, the first form should be used instead. Using the preceding rules without the marked extensions, any of the three comparison forms is reliable, given any input. (However, observe that the strings are quoted in all cases.)

Because the string comparison binary primaries, `=` and `!=`, have a higher precedence than any unary primary in the >4 argument case, unexpected results can occur if arguments are not properly prepared. For example, in

```
   test -d $1 -o -d $2
```

Last Revised 10 Sep 2001
If $1$ evaluates to a possible directory name of =, the first three arguments are considered a string comparison, which causes a syntax error when the second -d is encountered. One of the following forms prevents this; the second is preferred:

\[
\text{test } \{ \text{-d "$1" \} -o \{ -d "$2" \}}
\]

\[
\text{test -d "$1" || test -d "$2"}
\]

Also in the >4 argument case:

\[
\text{test "$1" = "bat" -a "$2" = "ball"}
\]

Syntax errors will occur if $1$ evaluates to ( or !. One of the following forms prevents this; the third is preferred:

\[
\text{test } "X$1" = "Xbat" -a "X$2" = "Xball"}
\]

\[
\text{test "$1" = "bat" && test "$2" = "ball"}
\]

\[
\text{test } "X$1" = "Xbat" && test "X$2" = "Xball"
\]

### EXAMPLES

In the if command examples, three conditions are tested, and if all three evaluate as true or successful, then their validities are written to the screen. The three tests are:

- if a variable set to 1 is greater than 0,
- if a variable set to 2 is equal to 2, and
- if the word "root" is included in the text file /etc/passwd.

```bash
/usr/bin/test
```

**EXAMPLE 1** Using /usr/bin/test

Perform a mkdir if a directory does not exist:

```bash
test ! -d tempdir && mkdir tempdir
```

Wait for a file to become non-readable:

```bash
while test -r thefile
do
  sleep 30
done
echo 'thefile is no longer readable'
```

Perform a command if the argument is one of three strings (two variations), using the open bracket version [ of the test command:

```bash
if [ "$1" = "pear" ] || [ "$1" = "grape" ] || [ "$1" = "apple" ]
then
  command
fi
case "$1" in
  pear|grape|apple) command;;
esac
```
The two forms of the test built-in follow the Bourne shell’s if example.

**EXAMPLE 2** Using the sh built-in

ZERO=0 ONE=1 TWO=2 ROOT=root

if [ $ONE -gt $ZERO ]
  [ $TWO -eq 2 ]
grep $ROOT /etc/passwd >&1 > /dev/null # discard output
then
  echo "$ONE is greater than 0, $TWO equals 2, and $ROOT is" \
  "a user-name in the password file"
else
  echo "At least one of the three test conditions is false"
fi

**EXAMPLE 3** Using the test built-in

Examples of the test built-in:

test `grep $ROOT /etc/passwd >&1` # discard output

echo $? # test for success
[ `grep nosuchname /etc/passwd >&1` ]
echo $? # test for failure

csh **EXAMPLE 4** Using the csh built-in

@ ZERO = 0; @ ONE = 1; @ TWO = 2; set ROOT = root
grep $ROOT /etc/passwd >&1 /dev/null # discard output # $status must be tested for immediately following grep
if ( "$status" == "0" && $ONE > $ZERO && $TWO == 2 ) then
  echo "$ONE is greater than 0, $TWO equals 2, and $ROOT is" \
  "a user-name in the password file"
endif

ksh **EXAMPLE 5** Using the ksh built-in

ZERO=0 ONE=1 TWO=$((ONE+ONE)) ROOT=root

if ((ONE > ZERO)) # arithmetical comparison
  [[ $TWO == 2 ]] # string comparison
  [ `grep $ROOT /etc/passwd >&1 /dev/null` ] # discard output
then
  echo "$ONE is greater than 0, $TWO equals 2, and $ROOT is" \
  "a user-name in the password file"
else
  echo "At least one of the three test conditions is false"
EXAMPLE 5 Using the ksh built-in \( \texttt{fi} \) (Continued)

ENVIRONMENT VARIABLES
See environ(5) for descriptions of the following environment variables that affect the execution of test: LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS
The following exit values are returned:

- \num{0} \textit{condition} evaluated to true.
- \num{1} \textit{condition} evaluated to false or \textit{condition} was missing.
- \num{>1} An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO
csh(1), ksh(1), sh(1), test(1B), attributes(5), environ(5), largefile(5)

NOTES
The \texttt{not-a-directory} alternative to the \texttt{-f} option is a transition aid for BSD applications and may not be supported in future releases.
# NAME
test – condition evaluation command

# SYNOPSIS
```
/usr/ucb/test expression
```

# DESCRIPTION
test evaluates the expression `expression` and, if its value is true, sets 0 (true) exit status; otherwise, a non-zero (false) exit status is set. test also sets a non-zero exit status if there are no arguments. When permissions are tested, the effective user ID of the process is used.

All operators, flags, and brackets (brackets used as shown in the second SYNOPSIS line) must be separate arguments to the test command; normally these items are separated by spaces.

## Primitives

The following primitives are used to construct `expression`:

- `-r filename` True if `filename` exists and is readable.
- `-w filename` True if `filename` exists and is writable.
- `-x filename` True if `filename` exists and is executable.
- `-f filename` True if `filename` exists and is a regular file. Alternatively, if `/usr/bin/sh` users specify `/usr/ucb` before `/usr/bin` in their `PATH` environment variable, then `test` will return true if `filename` exists and is (not-a-directory). This is also the default for `/usr/bin/csh` users.
- `-d filename` True if `filename` exists and is a directory.
- `-c filename` True if `filename` exists and is a character special file.
- `-b filename` True if `filename` exists and is a block special file.
- `-p filename` True if `filename` exists and is a named pipe (fifo).
- `-u filename` True if `filename` exists and its set-user-ID bit is set.
- `-g filename` True if `filename` exists and its set-group-ID bit is set.
- `-k filename` True if `filename` exists and its sticky bit is set.
- `-s filename` True if `filename` exists and has a size greater than zero.
- `-t [fdes ]` True if the open file whose file descriptor number is `fdes` (1 by default) is associated with a terminal device.
- `-z s1` True if the length of string `s1` is zero.
- `-n s1` True if the length of the string `s1` is non-zero.

`s1`  `s2` True if strings `s1` and `s2` are identical.

`s1`  `s2` True if strings `s1` and `s2` are not identical.

`s1` True if `s1` is not the null string.
n1 −eq n2  True if the integers n1 and n2 are algebraically equal. Any of the
comparisons −ne, −gt, −ge, −lt, and −le may be used in place of
−eq.

Operators  These primaries may be combined with the following operators:

!  Unary negation operator.
-a  Binary and operator.
-o  Binary or operator (−a has higher precedence than −o).
(expression)  Parentheses for grouping. Notice also that parentheses are
meaningful to the shell and, therefore, must be quoted.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

SEE ALSO  find(1), sh(1), attributes(5)

NOTES  The not-a-directory alternative to the −f option is a transition aid for BSD
applications and may not be supported in future releases.

If you test a file you own (the −r, −w, or −x tests), but the permission tested does
not have the owner bit set, a non-zero (false) exit status will be returned even though
the file may have the group or other bit set for that permission. The correct exit status
will be set if you are super-user.

The = and != operators have a higher precedence than the −r through −n operators,
and = and != always expect arguments; therefore, = and != cannot be used with the
−r through −n operators.

If more than one argument follows the −r through −n operators, only the first
argument is examined; the others are ignored, unless a −a or a −o is the second
argument.
NAME  test – condition evaluation command

SYNOPSIS  test expression
expression

DESCRIPTION  test evaluates the expression expression and if its value is true, sets a 0 (TRUE) exit status; otherwise, a non-zero (FALSE) exit status is set; test also sets a non-zero exit status if there are no arguments. When permissions are tested, the effective user ID of the process is used.

All operators, flags, and brackets (brackets used as shown in the second SYNOPSIS line) must be separate arguments to test. Normally these items are separated by spaces.

Primitives  The following primitives are used to construct expression:
- r filename True if filename exists and is readable.
- w filename True if filename exists and is writable.
- x filename True if filename exists and is executable.
- f filename True if filename exists and is a regular file.
- d filename True if filename exists and is a directory.
- c filename True if filename exists and is a character special file.
- b filename True if filename exists and is a block special file.
- p filename True if filename exists and is a named pipe (FIFO).
- u filename True if filename exists and its set-user-ID bit is set.
- g filename True if filename exists and its set-group-ID bit is set.
- k filename True if filename exists and its sticky bit is set.
- s filename True if filename exists and has a size greater than 0.
- t [ fdlist ] True if the open file whose file descriptor number is fdlist (1 by default) is associated with a terminal device.
- z s1 True if the length of string s1 is 0.
- n s1 True if the length of the string s1 is non-zero.
Operators
These primaries may be combined with the following operators:

\[ s1 = s2 \]
True if strings \( s1 \) and \( s2 \) are identical.

\[ s1 != s2 \]
True if strings \( s1 \) and \( s2 \) are not identical.

\[ s1 \]
True if \( s1 \) is not the null string.

\[ n1 -eq n2 \]
True if the integers \( n1 \) and \( n2 \) are algebraically equal. Any of the comparisons \(-\text{ne}, -\text{gt}, -\text{ge}, -\text{lt}, \text{ and } -\text{le}\) may be used in place of \(-\text{eq}\).

\[
\begin{align*}
&! \\
&-a \\
&-o
\end{align*}
\]
Unary negation operator.

Binary and operator.

Binary or operator (-a has higher precedence than -o).

\[ '(expression)' \]
Parentheses for grouping. Notice also that parentheses are meaningful to the shell and, therefore, must be quoted.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO
find(1), sh(1), attributes(5)

NOTES
If you test a file you own (the -r, -w, or -x tests), but the permission tested does not have the owner bit set, a non-zero (false) exit status will be returned even though the file may have the group or other bit set for that permission. The correct exit status will be set if you are super-user.

The = and != operators have a higher precedence than the -r through -n operators, and = and != always expect arguments; therefore, = and != cannot be used with the -r through -n operators.

If more than one argument follows the -r through -n operators, only the first argument is examined; the others are ignored, unless a -a or a -o is the second argument.
tftp is the user interface to the Internet TFTP (Trivial File Transfer Protocol), which allows users to transfer files to and from a remote machine. The remote host and optional port may be specified on the command line, in which case tftp uses host as the default host, and if specified, port as the default port, for future transfers. See the connect command below.

Once tftp is running, it issues the prompt tftp> and recognizes the following commands:

**Commands**

- **connect host-name [port]**
  - Set the host, and optionally port, for transfers. The TFTP protocol, unlike the FTP protocol, does not maintain connections between transfers; thus, the connect command does not actually create a connection, but merely remembers what host is to be used for transfers. You do not have to use the connect command; the remote host can be specified as part of the get or put commands.

- **mode transfer-mode**
  - Set the mode for transfers; transfer-mode may be one of ascii or binary. The default is ascii.

- **put filename**
- **put localfile remotefile**
- **put filename1 filename2 ... filenameN remote-directory**
  - Transfer a file, or a set of files, to the specified remote file or directory. The destination can be in one of two forms: a filename on the remote host if the host has already been specified, or a string of the form:

    host:filename

    to specify both a host and filename at the same time. If the latter form is used, the specified host becomes the default for future transfers. If the remote-directory form is used, the remote host is assumed to be running the UNIX system.

    The host can be a host name (see hosts(4) or ipnodes(4)) or an IPv4 or IPv6 address string (see inet(7P) or inet6(7P)). Since IPv6 addresses already contain "::", the host should be enclosed in square brackets when an IPv6 address is used. Otherwise, the first occurrence of a colon will be interpreted as the separator between the host and the filename. For example,

    [1080::8:800:200c:417A]:myfile

    Files may be written only if they already exist and are publicly writable. See in.tftpd(1M).

- **get filename**
- **get remotename localname**
- **get filename1 filename2 filename3 ... filenameN**
Get a file or set of files (three or more) from the specified remote sources. source can be in one of two forms: a filename on the remote host if the host has already been specified, or a string of the form:

host:filename

to specify both a host and filename at the same time. If the latter form is used, the last host specified becomes the default for future transfers. See the put command regarding specifying a host.

quit
Exit tftp. An EOF also exits.

verbose
Toggle verbose mode.

trace
Toggle packet tracing.

status
Show current status.

rexmt retransmission-timeout
Set the per-packet retransmission timeout, in seconds.

timeout total-transmission-timeout
Set the total transmission timeout, in seconds.

ascii
Shorthand for mode ascii.

binary
Shorthand for mode binary.

blksize transfer-blocksize
The value of the transfer blocksize option to negotiate with the server. A value of 0 disables the negotiation of this option.

srexmt server-retransmission-timeout
The value of the retransmission timeout option to request that the server uses. A value of 0 disables the negotiation of this option.

tsize
A toggle that sends the transfer size option to the server. By default, the option is not sent. The transfer size option is not sent with a write request when the transfer-mode is ascii.

? [ command-name ... ]
Print help information.

ATTRIBUTES
See attributes(5) for descriptions of the following attributes:
<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWtftp</td>
</tr>
</tbody>
</table>

**SEE ALSO**

in.tftpd(1M), hosts(4), ipnodes(4), attributes(5), inet(7P), inet6(7P)

Malkin, G. and Harkin, A. RFC 2347, **TFTP Option Extension**. The Internet Society. May 1998

Malkin, G. and Harkin, A. RFC 2348, **TFTP Blocksize Option**. The Internet Society. May 1998

Malkin, G. and Harkin, A. RFC 2349, **TFTP Timeout Interval and Transfer Size Options**. The Internet Society. May 1998


**NOTES**

The default *transfer-mode* is *ascii*. This differs from pre-SunOS 4.0 and pre-4.3BSD systems, so explicit action must be taken when transferring non-ASCII binary files such as executable commands.

Because there is no user-login or validation within the TFTP protocol, many remote sites restrict file access in various ways. Approved methods for file access are specific to each site, and therefore cannot be documented here.

When using the `get` command to transfer multiple files from a remote host, three or more files must be specified. If two files are specified, the second file is used as a local file.
The `time` utility invokes `utility` operand with `argument`, and writes a message to standard error that lists timing statistics for `utility`. The message includes the following information:

- The elapsed (real) time between invocation of `utility` and its termination.
- The User CPU time, equivalent to the sum of the `tms_utime` and `tms_cutime` fields returned by the `times()` function for the process in which `utility` is executed.
- The System CPU time, equivalent to the sum of the `tms_stime` and `tms_cstime` fields returned by the `times()` function for the process in which `utility` is executed.

When `time` is used as part of a pipeline, the times reported are unspecified, except when it is the sole command within a grouping command in that pipeline. For example, the commands on the left are unspecified; those on the right report on utilities `a` and `c`, respectively.

```
time a | b | c       { time a } | b | c
a | b | time c     a | b | (time c)
```

The following option is supported:

- `-p` Write the timing output to standard error in the following format:

  ```
  real %f
  user %f
  sys %f
  <real seconds>, <user seconds>, <system seconds>
  ```

The following operands are supported:

**utility** The name of the utility that is to be invoked.

**argument** Any string to be supplied as an argument when invoking `utility`.

The `time` utility returns exit status 127 if an error occurs so that applications can distinguish “failure to find a utility” from “invoked utility exited with an error indication.” The value 127 was chosen because it is not commonly used for other meanings; most utilities use small values for “normal error conditions” and the values above 128 can be confused with termination due to receipt of a signal. The value 126 was chosen in a similar manner to indicate that the utility could be found, but not invoked.

**EXAMPLE 1** Using The `time` Command

It is frequently desirable to apply `time` to pipelines or lists of commands. This can be done by placing pipelines and command lists in a single file; this file can then be invoked as a utility, and the `time` applies to everything in the file.

Alternatively, the following command can be used to apply `time` to a complex command:
EXAMPLE 1 Using The time Command  (Continued)

time sh -c 'complex-command-line'

EXAMPLE 2 Using time In The csh Shell

The following two examples show the differences between the csh version of time and the version in /usr/bin/time. These examples assume that csh is the shell in use.

eexample% time find / -name csh.1 -print
/usr/share/man/man1/csh.1
95.0u 692.0s 1:17:52 16% 0+0k 0+0io 0pf+0w
See csh(1) for an explanation of the format of time output.

eexample% /usr/bin/time find / -name csh.1 -print
/usr/share/man/man1/csh.1
real 1:23:31.5
user 1:33.2
sys 11:28.2

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of time: LC_CTYPE, LC_MESSAGES, LC_NUMERIC, NLSPATH, and PATH.

EXIT STATUS

If utility is invoked, the exit status of time will be the exit status of utility; otherwise, the time utility will exit with one of the following values:

1−125 An error occurred in the time utility.

126 utility was found but could not be invoked.

127 utility could not be found.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO
csh(1), shell_builtins(1), timex(1), times(2), attributes(5), environ(5)

NOTES

When the time command is run on a multiprocessor machine, the total of the values printed for user and sys can exceed real. This is because on a multiprocessor machine it is possible to divide the task between the various processors.

When the command being timed is interrupted, the timing values displayed may not always be accurate.
Elapsed time is accurate to the second, while the CPU times are measured to the 100th second. Thus the sum of the CPU times can be up to a second larger than the elapsed time.
times(1)

NAME
times – shell built-in function to report time usages of the current shell

sh
times

ksh
times

sh
Print the accumulated user and system times for processes run from the shell.

ksh
Print the accumulated user and system times for the shell and for processes run from
the shell.

On this man page, ksh(1) commands that are preceded by one or two * (asterisks) are
treated specially in the following ways:
1. Variable assignment lists preceding the command remain in effect when the
command completes.
2. I/O redirections are processed after variable assignments.
3. Errors cause a script that contains them to abort.
4. Words, following a command preceded by ** that are in the format of a variable
assignment, are expanded with the same rules as a variable assignment. This
means that tilde substitution is performed after the = sign and word splitting and
file name generation are not performed.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO ksh(1), sh(1), time(1), attributes(5)
**NAME**

`timex` – time a command; report process data and system activity

**SYNOPSIS**

```
timex [-o] [-p [-fhkmrt]] [-s] `command`
```

**DESCRIPTION**

The given `command` is executed; the elapsed time, user time and system time spent in execution are reported in seconds. Optionally, process accounting data for the `command` and all its children can be listed or summarized, and total system activity during the execution interval can be reported.

The output of `timex` is written on standard error.

**OPTIONS**

The following options are supported:

- `-o` Report the total number of blocks read or written and total characters transferred by `command` and all its children. This option works only if the process accounting software is installed.

- `-p` List process accounting records for `command` and all its children. This option works only if the process accounting software is installed. Suboptions `-f`, `-h`, `-k`, `-m`, `-r`, and `-t` modify the data items reported. The options are as follows:

  - `-f` Print the `fork(2)/exec(2)` flag and system exit status columns in the output.

  - `-h` Instead of mean memory size, show the fraction of total available CPU time consumed by the process during its execution. This “hog factor” is computed as (total CPU time)/(elapsed time).

  - `-k` Instead of memory size, show total kcore-minutes.

  - `-m` Show mean core size (the default).

  - `-r` Show CPU factor (user time/(system-time + user-time)).

  - `-t` Show separate system and user CPU times. The number of blocks read or written and the number of characters transferred are always reported.

- `-s` Report total system activity (not just that due to `command`) that occurred during the execution interval of `command`. All the data items listed in `sar(1)` are reported.

**EXAMPLES**

**EXAMPLE 1** Examples of `timex`.

A simple example:
```
example$ timex -ops sleep 60
```

A terminal session of arbitrary complexity can be measured by timing a sub-shell:
```
example$ timex -opskmt sh
session commands
```

EOT
EXAMPLE 1 Examples of timex. (Continued)

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWaccu</td>
</tr>
</tbody>
</table>

SEE ALSO  sar(1), time(1), exec(2), fork(2), times(2), attributes(5)

NOTES  Process records associated with command are selected from the accounting file /var/adm/pacct by inference, since process genealogy is not available. Background processes having the same user ID, terminal ID, and execution time window will be spuriously included.
The `tip` utility establishes a full-duplex terminal connection to a remote host. Once the connection is established, a remote session using `tip` behaves like an interactive session on a local terminal.

The `remote` file contains entries describing remote systems and line speeds used by `tip`.

Each host has a default baud rate for the connection, or you can specify a speed with the `-speed-entry` command line argument.

When `phone-number` is specified, `tip` looks for an entry in the `remote` file of the form:

```
tip -speed-entry
```

When `tip` finds such an entry, it sets the connection speed accordingly. If it finds no such entry, `tip` interprets `-speed-entry` as if it were a system name, resulting in an error message.

If you omit `-speed-entry`, `tip` uses the `tip0` entry to set a speed for the connection.

When `device` is specified, `tip` attempts to open that device, but will do so using the access privileges of the user, rather than `tip`'s usual access privileges (`setuid uucp`). The user must have read/write access to the device. The `tip` utility interprets any character string beginning with the slash character (`/`) as a device name.

When establishing the connection, `tip` sends a connection message to the remote system. The default value for this message can be found in the `remote` file.

When `tip` attempts to connect to a remote system, it opens the associated device with an exclusive-open `ioctl(2)` call. Thus, only one user at a time may access a device. This is to prevent multiple processes from sampling the terminal line. In addition, `tip` honors the locking protocol used by `uucp(1C)`.

When `tip` starts up, it reads commands from the file `.tiprc` in your home directory.

```
-v         Display commands from the `.tiprc` file as they are executed.
```

Typed characters are normally transmitted directly to the remote machine, which does the echoing as well.

At any time that `tip` prompts for an argument (for example, during setup of a file transfer), the line typed may be edited with the standard erase and kill characters. A null line in response to a prompt, or an interrupt, aborts the dialogue and returns you to the remote machine.

A tilde (`~`) appearing as the first character of a line is an escape signal which directs `tip` to perform some special action. `tip` recognizes the following escape sequences:
Drop the connection and exit (you may still be logged in on the remote machine). Note: If you rlogin and then run tip on the remote host, you must type `~`~. (tilde tilde dot) to end the tip session. If you type `~`. (tilde dot), it terminates the rlogin.

`-c [name]` Change directory to `name`. No argument implies change to your home directory.

`-!` Escape to an interactive shell on the local machine. Exiting the shell returns you to tip.

`->` Copy file from local to remote.

`<-` Copy file from remote to local.

`-p from [ to ]` Send a file to a remote host running the UNIX system. When you use the put command, the remote system runs the command string

```
cat > to
```

while tip sends it the `from` file. If the `to` file is not specified, the `from` file name is used. This command is actually a UNIX-system-specific version of the `->` command.

`-t from [ to ]` Take a file from a remote host running the UNIX system. As in the put command the `to` file defaults to the `from` file name if it is not specified. The remote host executes the command string

```
cat from ; echo ^A
```
to send the file to tip.

`-|` Pipe the output from a remote command to a local process. The command string sent to the local system is processed by the shell.

`-C` Connect a program to the remote machine. The command string sent to the program is processed by the shell. The program inherits file descriptors 0 as remote line input, 1 as remote line output, and 2 as tty standard error.

`-$` Pipe the output from a local process to the remote host. The command string sent to the local system is processed by the shell.

`-#` Send a BREAK to the remote system.

`s` Set a variable (see the discussion below).

`-^Z` Stop tip. Only available when run under a shell that supports job control, such as the C shell.

`-^Y` Stop only the “local side” of tip. Only available when run under a shell that supports job control, such as the C shell. The “remote side” of tip, that is, the side that displays output from the remote host, is left running.
Get a summary of the tilde escapes.

Copying files requires some cooperation on the part of the remote host. When a \texttt{\textasciitilde\textasciitilde} or \texttt{\textasciitilde<} escape is used to send a file, \texttt{tip} prompts for a file name (to be transmitted or received) and a command to be sent to the remote system, in case the file is being transferred from the remote system. While \texttt{tip} is transferring a file, the number of lines transferred will be continuously displayed on the screen. A file transfer may be aborted with an interrupt.

### Auto-call Units

\texttt{tip} may be used to dial up remote systems using a number of auto-call unit’s (ACUs). When the remote system description contains the \texttt{\textasciitilde du} capability, \texttt{tip} uses the call-unit (\texttt{cu}), ACU type (\texttt{at}), and phone numbers (\texttt{pn}) supplied. Normally, \texttt{tip} displays verbose messages as it dials.

Depending on the type of auto-dialer being used to establish a connection, the remote host may have garbage characters sent to it upon connection. The user should never assume that the first characters typed to the foreign host are the first ones presented to it. The recommended practice is to immediately type a \texttt{kill} character upon establishing a connection (most UNIX systems either support \texttt{@} or Control-U as the initial kill character).

\texttt{tip} currently supports the Ventel MD-212+ modem and DC Hayes-compatible modems.

When \texttt{tip} initializes a Hayes-compatible modem for dialing, it sets up the modem to auto-answer. Normally, after the conversation is complete, \texttt{tip} drops DTR, which causes the modem to “hang up.”

Most modems can be configured so that when DTR drops, they re-initialize themselves to a preprogrammed state. This can be used to reset the modem and disable auto-answer, if desired.

Additionally, it is possible to start the phone number with a Hayes \texttt{\$} command so that you can configure the modem before dialing. For example, to disable auto-answer, set up all the phone numbers in \texttt{/etc/remote} using something like \texttt{pn=S0=0DT5551212}. The \texttt{S0=0} disables auto-answer.

### Remote Host Description

Descriptions of remote hosts are normally located in the system-wide file \texttt{/etc/remote}. However, a user may maintain personal description files (and phone numbers) by defining and exporting the \texttt{REMOTE} shell variable. The \texttt{remote} file must be readable by \texttt{tip}, but a secondary file describing phone numbers may be maintained readable only by the user. This secondary phone number file is \texttt{/etc/phones}, unless the shell variable \texttt{PHONES} is defined and exported. The phone number file contains lines of the form:

\texttt{system-name phone-number}
Each phone number found for a system is tried until either a connection is established, or an end of file is reached. Phone numbers are constructed from '0123456789-=*', where the '=' and '*' are used to indicate a second dial tone should be waited for (ACU dependent).

```
-escapemaintains a set of variables which are used in normal operation. Some of these variables are read-only to normal users (root is allowed to change anything of interest). Variables may be displayed and set through the -s escape. The syntax for variables is patterned after vi(1) and mail(1). Supplying all as an argument to the -s escape displays all variables that the user can read. Alternatively, the user may request display of a particular variable by attaching a ? to the end. For example, '-s escape?' displays the current escape character.

Variables are numeric (num), string (str), character (char), or Boolean (bool) values. Boolean variables are set merely by specifying their name. They may be reset by prepending a ! to the name. Other variable types are set by appending an = and the value. The entire assignment must not have any blanks in it. A single set command may be used to interrogate as well as set a number of variables.

Variables may be initialized at run time by placing set commands (without the -s prefix) in a .tiprc file in one's home directory. The -v option makes tip display the sets as they are made. Comments preceded by a # sign can appear in the .tiprc file.

Finally, the variable names must either be completely specified or an abbreviation may be given. The following list details those variables known to tip.

- **beautify** (bool) Discard unprintable characters when a session is being scripted; abbreviated be. If the nb capability is present, beautify is initially set to off. Otherwise, beautify is initially set to on.
- **baudrate** (num) The baud rate at which the connection was established; abbreviated ba. If a baud rate was specified on the command line, baudrate is initially set to the specified value. Or, if the br capability is present, baudrate is initially set to the value of that capability. Otherwise, baudrate is set to 300 baud. Once tip has been started, baudrate can only be changed by the super-user.
- **dialtimeout** (num) When dialing a phone number, the time (in seconds) to wait for a connection to be established; abbreviated dial. dialtimeout is initially set to 60 seconds, and can only be changed by the super-user.
- **disconnect** (str) The string to send to the remote host to disconnect from it; abbreviated di. If the di capability is present, disconnect is initially set to the value of that capability. Otherwise, disconnect is set to a null string ("").
- **echocheck** (bool) Synchronize with the remote host during file transfer by waiting for the echo of the last character transmitted; abbreviated
If the ec capability is present, echocheck is initially set to on. Otherwise, echocheck is initially set to off.

eofread (str) The set of characters which signify an end-of-transmission during a -< file transfer command; abbreviated eofr. If the ie capability is present, eofread is initially set to the value of that capability. Otherwise, eofread is set to a null string ("").

eofwrite (str) The string sent to indicate end-of-transmission during a -> file transfer command; abbreviated eofw. If the oe capability is present, eofread is initially set to the value of that capability. Otherwise, eofread is set to a null string ("").

eol (str) The set of characters which indicate an end-of-line. tip will recognize escape characters only after an end-of-line. If the el capability is present, eol is initially set to the value of that capability. Otherwise, eol is set to a null string ("").

escape (char) The command prefix (escape) character; abbreviated es. If the es capability is present, escape is initially set to the value of that capability. Otherwise, escape is set to ‘~’.

etimeout (num) The amount of time, in seconds, that tip should wait for the echo-check response when echocheck is set; abbreviated et. If the et capability is present, etimeout is initially set to the value of that capability. Otherwise, etimeout is set to 10 seconds.

exceptions (str) The set of characters which should not be discarded due to the beautification switch; abbreviated ex. If the ex capability is present, exceptions is initially set to the value of that capability. Otherwise, exceptions is set to '\t\n\f\b'.

force (char) The character used to force literal data transmission; abbreviated fo. If the fo capability is present, force is initially set to the value of that capability. Otherwise, force is set to \377 (which disables it).

framesize (num) The amount of data (in bytes) to buffer between file system writes when receiving files; abbreviated fr. If the fs capability is present, framesize is initially set to the value of that capability. Otherwise, framesize is set to 1024.

halfduplex (bool) Do local echoing because the host is half-duplex; abbreviated hdx. If the hd capability is present, halfduplex is initially set to on. Otherwise, halfduplex is initially set to off.

hardwareflow (bool) Do hardware flow control; abbreviated hf. If the hf capability is present, hardwareflow is initially set to on. Otherwise, hardwareflowcontrol is initially set to off.
### host

The name of the host to which you are connected; abbreviated ho. `host` is permanently set to the name given on the command line or in the `HOST` environment variable.

### localecho

A synonym for `halfduplex`; abbreviated `le`.

### log

The name of the file to which to log information about outgoing phone calls. `log` is initially set to `/var/adm/aculog`, and can only be inspected or changed by the super-user.

### parity

The parity to be generated and checked when talking to the remote host; abbreviated `par`. The possible values are:

- **none**
  - Parity is not checked on input, and the parity bit is set to zero on output.
- **zero**
  - Parity is not checked on input, and the parity bit is set to one on output.
- **one**
  - Parity is not checked on input, and the parity bit is set to one on output.
- **even**
  - Even parity is checked for on input and generated on output.
- **odd**
  - Odd parity is checked for on input and generated on output.

If the `pa` capability is present, `parity` is initially set to the value of that capability; otherwise, `parity` is set to `none`.

### phones

The file in which to find hidden phone numbers. If the environment variable `PHONES` is set, `phones` is set to the value of `PHONES`. Otherwise, `phones` is set to `/etc/phones`. The value of `phones` cannot be changed from within `tip`.

### prompt

The character which indicates an end-of-line on the remote host; abbreviated `pr`. This value is used to synchronize during data transfers. The count of lines transferred during a file transfer command is based on receipt of this character. If the `pr` capability is present, `prompt` is initially set to the value of that capability. Otherwise, `prompt` is set to `\n`.

### raise

Upper case mapping mode; abbreviated `ra`. When this mode is enabled, all lower case letters will be mapped to upper case by `tip` for transmission to the remote machine. If the `ra` capability is present, `raise` is initially set to `on`. Otherwise, `raise` is initially set to `off`.

### raisechar

The input character used to toggle upper case mapping mode; abbreviated `rc`. If the `rc` capability is present, `raisechar` is initially set to the value of that capability. Otherwise, `raisechar` is set to `\377` (which disables it).
<table>
<thead>
<tr>
<th>Environment Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>rawftp</code></td>
<td>(bool) Send all characters during file transfers; do not filter non-printable characters, and do not do translations like \n to \r. Abbreviated raw. If the rw capability is present, rawftp is initially set to on. Otherwise, rawftp is initially set to off.</td>
</tr>
<tr>
<td><code>record</code></td>
<td>(str) The name of the file in which a session script is recorded; abbreviated rec. If the re capability is present, record is initially set to the value of that capability. Otherwise, record is set to tip.record.</td>
</tr>
<tr>
<td><code>remote</code></td>
<td>The file in which to find descriptions of remote systems. If the environment variable REMOTE is set, remote is set to the value of REMOTE. Otherwise, remote is set to /etc/remote. The value of remote cannot be changed from within tip.</td>
</tr>
<tr>
<td><code>script</code></td>
<td>(bool) Session scripting mode; abbreviated sc. When script is on, tip will record everything transmitted by the remote machine in the script record file specified in record. If the beautify switch is on, only printable ASCII characters will be included in the script file (those characters between 040 and 0177). The variable exceptions is used to indicate characters which are an exception to the normal beautification rules. If the sc capability is present, script is initially set to on. Otherwise, script is initially set to off.</td>
</tr>
<tr>
<td><code>tabexpand</code></td>
<td>(bool) Expand TAB characters to SPACE characters during file transfers; abbreviated tab. When tabexpand is on, each tab is expanded to eight SPACE characters. If the tb capability is present, tabexpand is initially set to on. Otherwise, tabexpand is initially set to off.</td>
</tr>
<tr>
<td><code>tandem</code></td>
<td>(bool) Use XON/XOFF flow control to limit the rate that data is sent by the remote host; abbreviated ta. If the nt capability is present, tandem is initially set to off. Otherwise, tandem is initially set to on.</td>
</tr>
<tr>
<td><code>verbose</code></td>
<td>(bool) Verbose mode; abbreviated verb; When verbose mode is enabled, tip prints messages while dialing, shows the current number of lines transferred during a file transfer operations, and more. If the nv capability is present, verbose is initially set to off. Otherwise, verbose is initially set to on.</td>
</tr>
<tr>
<td><code>SHELL</code></td>
<td>(str) The name of the shell to use for the ~! command; default value is /bin/sh, or taken from the environment.</td>
</tr>
<tr>
<td><code>HOME</code></td>
<td>(str) The home directory to use for the ~c command. Default value is taken from the environment.</td>
</tr>
</tbody>
</table>

**EXAMPLES**

**EXAMPLE 1** Using the tip command

An example of the dialog used to transfer files is given below.
EXAMPLE 1 Using the tip command (Continued)

arpa% tip monet
[connected]
...(assume we are talking to a UNIX system)...
ucbmonet login: sam
Password:
monet% cat sylvester.c
-> Filename: sylvester.c
32 lines transferred in 1 minute 3 seconds
monet%
monet% ~< Filename: reply.c
List command for remote host: cat reply.c
65 lines transferred in 2 minutes
monet%
...(or, equivalently)...
monet% -p sylvester.c
...(actually echoes as -[put] sylvester.c)...
32 lines transferred in 1 minute 3 seconds
monet%
monet% -t reply.c
...(actually echoes as -[take] reply.c)...
65 lines transferred in 2 minutes
monet%
...(to print a file locally)...
monet% ^D
[EOF]
...(back on the local system)...

ENVIRONMENT VARIABLES
The following environment variables are read by tip.
REMOTE The location of the remote file.
PHONES The location of the file containing private phone numbers.
HOST A default host to connect to.
HOME One's log-in directory (for chdirs).
SHELL The shell to fork on a '-!' escape.

FILES
/etc/phones
/etc/remote
/var/spool/locks/LCK. *
    lock file to avoid conflicts with UUCP
/var/adm/aculog
    file in which outgoing calls are logged
~/.tiprc
    initialization file
See attributes(5) for descriptions of the following attributes:

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

There are two additional variables, chardelay and linedelay, that are currently not implemented.

SEE ALSO

cu(1C), mail(1), uucp(1C), vi(1), ioctl(2), attributes(5)

BUGS

There are two additional variables, chardelay and linedelay, that are currently not implemented.
tnfdump(1)

NAME
tnfdump – convert binary TNF file to ASCII

SYNOPSIS
tnfdump [-r] [-x] tnf_file...

DESCRIPTION
The tnfdump utility converts the specified binary TNF trace files to ASCII. The ASCII output can be used to do performance analysis. The default mode (without the -r option) prints all the event records (that were generated by TNF_PROBE(3TNF)) and the event descriptor records only. It also orders the events by time.

OPTIONS
The following option is supported:
- r  Does a raw conversion of TNF to ASCII. The output is a literal translation of the binary TNF file and includes all the records in the file. This output is useful only if you have a good understanding of TNF. A sample output is listed in EXAMPLES below.
- x  Prints all TNF unsigned type argument values in hexadecimal format instead of decimal format.

RETURN VALUES
tnfdump returns 0 on successful exit.

EXAMPLES
EXAMPLE 1 Converting a file into ASCII

To convert the file /tmp/trace-2130 into ASCII, use the tnfdump command and the name of the binary trace file. Be aware that the tnf dump output goes to stdout by default.

example% tnf dump /tmp/trace-2130

probe tnf_name: "inloop" tnf_string: "keys cookie main loop;\n    file cookie2.c; line 50; sunw%debug in the loop"

probe tnf_name: "end" tnf_string: "keys cookie main end;\n    file cookie2.c; line 41; sunw%debug exiting program"

<table>
<thead>
<tr>
<th>Elapsed (ms)</th>
<th>Delta (ms)</th>
<th>PID</th>
<th>LWID</th>
<th>TID</th>
<th>CPU</th>
<th>Probe Name</th>
<th>Description ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000000</td>
<td>0.000000</td>
<td>8792</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>inloop</td>
<td></td>
</tr>
<tr>
<td>0.339000</td>
<td>0.339000</td>
<td>8792</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>inloop</td>
<td></td>
</tr>
<tr>
<td>0.350500</td>
<td>0.011500</td>
<td>8792</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>inloop</td>
<td></td>
</tr>
<tr>
<td>0.359500</td>
<td>0.009000</td>
<td>8792</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>inloop</td>
<td></td>
</tr>
<tr>
<td>0.369500</td>
<td>0.010000</td>
<td>8792</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>inloop</td>
<td></td>
</tr>
<tr>
<td>7775.969500</td>
<td>7775.600000</td>
<td>8792</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>inloop</td>
<td></td>
</tr>
<tr>
<td>7776.016000</td>
<td>0.046500</td>
<td>8792</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>inloop</td>
<td></td>
</tr>
<tr>
<td>7776.025000</td>
<td>0.009000</td>
<td>8792</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>inloop</td>
<td></td>
</tr>
<tr>
<td>7776.034000</td>
<td>0.009000</td>
<td>8792</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>inloop</td>
<td></td>
</tr>
</tbody>
</table>

1548 man pages section 1: User Commands • Last Revised 22 Jan 2001
EXAMPLE 1 Converting a file into ASCII  (Continued)

All probes that are encountered during execution have a description of it printed out. The description is one per line prefixed by the keyword 'probe'. The name of the probe is in double quotes after the keyword 'tnf_name'. The description of this probe is in double quotes after the keyword 'tnf_string'.

A heading is printed after all the description of the probes are printed. The first column gives the elapsed time in milli-seconds since the first event. The second column gives the elapsed time in milli-seconds since the previous event. The next four columns are the process id, lwp id, thread id, and cpu number. The next column is the name of the probe that generated this event. This can be matched to the probe description explained above. The last column is the data that the event contains, formatted as arg_name_n (see T NF_PROBE(3TNF)) followed by a colon and the value of that argument. The format of the value depends on its type. tnf_opaque arguments are printed in hexadecimal. All other integers are printed in decimal. Strings are printed in double quotes and user-defined records are enclosed in braces '{ }'. The first field of a user defined record indicates its TNF type (see T NF_DECLARE_RECORD(3TNF)). The rest of the fields are the members of the record.

A '-' in any column indicates that there is no data for that particular column.

EXAMPLE 2 To do a raw conversion of a file into ASCII

To do a raw conversion of the file /tmp/trace-4000 into ASCII, use:

e xample$ tnf dump -r /tmp/trace-4000

The output will look like the following:

0x10e00 : {
    tnf_tag 0x109c0 tnf_block_header
    generation 1
    bytes_valid 320
    A_lock 0
    B_lock 0
    next_block 0x0
}

0x10e10 : {
    tnf_tag 0x10010 probe1
    tnf_tag_arg 0x10e24 <tnf_sched_rec>
    time_delta 128
}
EXAMPLE 2  To do a raw conversion of a file into ASCII  

(Continued)

    test_ulong 4294967295
test_long -1
}
0x10e24 : {
    tnf_tag 0x10cf4 tnf_sched_rec
tid 0
lwpid 1
pid 13568
time_base 277077875828500
}
0x10e3c : {
    tnf_tag 0x11010 probe2
    tnf_tag_arg 0x10e24 <tnf_sched_rec>
time_delta 735500
test_str 0x10e48 "string1"
}
0x10e48 : {
    tnf_tag 0x1072c tnf_string
    tnf_self_size 16
    chars "string1"
}
0x10e58 : {
    tnf_tag 0x110ec probe3
    tnf_tag_arg 0x10e24 <tnf_sched_rec>
time_delta 868000
test_ulonglong 18446744073709551615
test_longlong -1
test_float 3.142857
}
...  
0x110ec : {
    tnf_tag 0x10030 tnf_probe_type
    tnf_tag_code 42
    tnf_name 0x1110c "probe3"
    tnf_properties 0x1111c <tnf_properties>
    tnf_slot_types 0x11130 <tnf_slot_types>
    tnf_type_size 32
    tnf_slot_names 0x111c4 <tnf_slot_names>
    tnf_string 0x11268 "keys targdebug main;\n    file targdebug.c;line 61;"
}
0x1110c : {
    tnf_tag 0x10068 tnf_name
    tnf_self_size 16
    chars "probe3"
}
0x1111c : {
    tnf_tag 0x100b4 tnf_properties
    tnf_self_size 20
    0x101a0 tnf_tagged
    1 0x101c4 tnf_struct
    2 0x10b84 tnf_tag_arg
EXAMPLE 2  To do a raw conversion of a file into ASCII  

(Continued)

```
0x11130  : {
  tnf_tag 0x10210  tnf_slot_types
  tnf_self_size 20
  0 0x10bd0  tnf_probe_event
  1 0x10c20  tnf_time_delta
  2 0x1114c  tnf_uint64
  3 0x10d54  tnf_int64
  4 0x11188  tnf_float32
}
```

The first number is the file offset of the record. The record is enclosed in braces ‘{}’.
The first column in a record is the slot name (for records whose fields do not have
names, it is the type name). The second column in the record is the value of that slot if
it is a scalar (only scalars that are of type tnf_opaque are printed in hex), or the offset
of the record if it is a reference to another record.

The third column in a record is optional. It does not exist for scalar slots of records. If
it exists, the third column is a type name with or without angle brackets, or a string in
double quotes. Unadorned names indicate a reference to the named metatag record
(that is, a reference to a record with that name in the tnf_name field). Type names in
angled brackets indicate a reference to a record that is an instance of that type (that is,
a reference to a record with that name in the tnf_tag field). The content of strings are
printed out in double quotes at the reference site.

Records that are arrays have their array elements follow the header slots, and are
numbered 0, 1, 2, and so on, except strings where the string is written as the ‘chars’
(pseudo-name) slot.

Records that are events (generated by TNF_PROBE(3TNF)) will have a slot name of
tnf_tag_arg as their second field which is a reference to the schedule record.
Schedule records describe more information about the event like the thread-id,
process-id, and the time_base. The time_delta of an event can be added to the
time_base of the schedule record that the event references, to give an absolute time.
This time is expressed as nanoseconds since some arbitrary time in the past (see
gethrtime(3C)).

EXAMPLE 3 Printing TNF unsigned arguments in hexadecimal

To print TNF unsigned arguments in hexadecimal for the file /tmp/trace-2192,
use:

```
example$ tnfdump -x /tmp/trace-2192
```

The output will look like the following:

```
probe  tnf_name: "start" tnf_string: "keys cookie main; file test17.c;line 20;sunw%debug starting main"
probe  tnf_name: "inloop" tnf_string: "keys cookie main
```

User Commands  1551
EXAMPLE 3 Printing TNF unsigned arguments in hexadecimal  (Continued)

<table>
<thead>
<tr>
<th>Elapsed (ms)</th>
<th>Delta (ms)</th>
<th>PID</th>
<th>LWID</th>
<th>TID</th>
<th>CPU</th>
<th>Probe</th>
<th>Data/Description ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000000</td>
<td>0.000000</td>
<td>6280</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>start</td>
<td></td>
</tr>
<tr>
<td>2455.211311</td>
<td>2455.211311</td>
<td>6280</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>inloop</td>
<td>loop_count: 0x0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>total_iterations: 0x0</td>
</tr>
<tr>
<td>2455.215768</td>
<td>0.004457</td>
<td>6280</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>inloop</td>
<td>loop_count: 0x1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>total_iterations: 0x1</td>
</tr>
<tr>
<td>2455.217041</td>
<td>0.001273</td>
<td>6280</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>inloop</td>
<td>loop_count: 0x2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>total_iterations: 0x2</td>
</tr>
<tr>
<td>2455.218285</td>
<td>0.001244</td>
<td>6280</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>inloop</td>
<td>loop_count: 0x3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>total_iterations: 0x3</td>
</tr>
<tr>
<td>2455.219600</td>
<td>0.001315</td>
<td>6280</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>inloop</td>
<td>loop_count: 0x4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>total_iterations: 0x4</td>
</tr>
<tr>
<td>4058.815125</td>
<td>1603.595525</td>
<td>6280</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>inloop</td>
<td>loop_count: 0x0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>total_iterations: 0x5</td>
</tr>
<tr>
<td>4058.818699</td>
<td>0.003574</td>
<td>6280</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>inloop</td>
<td>loop_count: 0x1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>total_iterations: 0x6</td>
</tr>
<tr>
<td>4058.819931</td>
<td>0.001232</td>
<td>6280</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>inloop</td>
<td>loop_count: 0x2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>total_iterations: 0x7</td>
</tr>
<tr>
<td>4058.821264</td>
<td>0.001333</td>
<td>6280</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>inloop</td>
<td>loop_count: 0x3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>total_iterations: 0x8</td>
</tr>
<tr>
<td>4058.822520</td>
<td>0.001256</td>
<td>6280</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>inloop</td>
<td>loop_count: 0x4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>total_iterations: 0x9</td>
</tr>
<tr>
<td>4058.823781</td>
<td>0.001261</td>
<td>6280</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>inloop</td>
<td>loop_count: 0x5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>total_iterations: 0xa</td>
</tr>
<tr>
<td>4058.825037</td>
<td>0.001256</td>
<td>6280</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>inloop</td>
<td>loop_count: 0x6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>total_iterations: 0xb</td>
</tr>
<tr>
<td>13896.655450</td>
<td>9837.830413</td>
<td>6280</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>final</td>
<td>loop_count16: 0x258</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>total_iterations8: 0xb0</td>
</tr>
</tbody>
</table>

Notice that the loop_count and the total_iterations are TNF unsigned arguments. Their values are printed in hexadecimal when requested by option -x.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWtnfd</td>
</tr>
</tbody>
</table>

SEE ALSO

prex(1), gethrtime(3C), TNF_DECLARE_RECORD(3TNF), TNF_PROBE(3TNF), tnf_process_disable(3TNF), attributes(5)
tnfextract – extract kernel probes output into a trace file

**SYNOPSIS**

```
 tnfextract [-d dumpfile -n namelist] tnf_file
```

**DESCRIPTION**

The **tnfextract** utility collects kernel trace output from an in-core buffer in the Solaris kernel, or from the memory image of a crashed system, and generates a binary TNF trace file like those produced directly by user programs being traced.

Either both or neither of the `-d` and `-n` options must be specified. If neither is specified, trace output is extracted from the running kernel. If both are specified, the `-d` argument names the file containing the (crashed) system memory image, and the `-n` argument names the file containing the symbol table for the system memory image.

The TNF trace file `tnf_file` produced is exactly the same size as the in-core buffer; it is essentially a snapshot of that buffer. It is legal to run **tnfextract** while kernel tracing is active, i.e., while the in-core buffer is being written. **tnfextract** insures that the output file it generates is low-level consistent, i.e., that only whole probes are written out, and that internal data structures in the buffer are not corrupted because the buffer is being concurrently written.

The TNF trace file generated is suitable as input to **tnfdump**(1), which will generate an ASCII file.

**OPTIONS**

The following options are supported:

- `-d dumpfile` Use `dumpfile` as the system memory image, instead of the running kernel. The `dumpfile` is normally the path name of a file generated by the `savecore` utility.

- `-n namelist` Use `namelist` as the file containing the symbol table information for the given `dumpfile`.

**OPERANDS**

The following operand is supported:

```
 tnf_file       output file generated by tnfextract based on kernel trace output from an in-core buffer in the Solaris kernel.
```

**EXAMPLES**

**EXAMPLE 1** Extracting probes from a running kernel

Extract probes from the running kernel into `ktrace.out`:

```
examp1$ tnfextract ktrace.out
```

**EXAMPLE 2** Extracting probes from a kernel crash dump

Extract probes from a kernel crash dump into `ktrace.out`:

```
examp1$ tnfextract -d /var/crash/`uname -n`/vmcore.0 \\ 
    -n /var/crash/`uname -n`/unix.0 ktrace.out
```

**EXIT STATUS**

The following exit values are returned:
tnfxtract(1)

0  Successful completion.
>0  An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWtnfc (32-bit)</td>
</tr>
<tr>
<td></td>
<td>SUNWtnfcx (64-bit)</td>
</tr>
</tbody>
</table>

SEE ALSO
prex(1), tnfdump(1), savecore(1M), tnf_kernel_probes(4), attributes(5)
The touch utility sets the access and modification times of each file. The file operand is created if it does not already exist.

The time used can be specified by -t time, by the corresponding time fields of the file referenced by -r ref_file, or by the date_time operand. If none of these are specified, touch uses the current time (the value returned by the time(2) function).

If neither the -a nor -m options are specified, touch updates both the modification and access times.

A user with write access to a file, but who is not the owner of the file or a super-user, can change the modification and access times of that file only to the current time. Attempts to set a specific time with touch will result in an error.

The settime utility is equivalent to touch -c [date_time] file.

touch

The following options are supported for the touch utility:

- **-a** Changes the access time of file. Does not change the modification time unless -m is also specified.

- **-c** Does not create a specified file if it does not exist. Does not write any diagnostic messages concerning this condition.

- **-m** Changes the modification time of file. Does not change the access time unless -a is also specified.

- **-r ref_file** Uses the corresponding times of the file named by ref_file instead of the current time.

- **-t time** Uses the specified time instead of the current time. time will be a decimal number of the form:

  \([\{CC\}\{YY\}]MMDDhhmm\.SS\) where each two digits represent the following:

  \(MM\) The month of the year [01-12].

  \(DD\) The day of the month [01-31].

  \(hh\) The hour of the day [00-23].

  \(mm\) The minute of the hour [00-59].

  \(CC\) The first two digits of the year.

  \(YY\) The second two digits of the year.
touch(1)

SS The second of the minute [00-61].

Both CC and YY are optional. If neither is given, the current year will be assumed. If YY is specified, but CC is not, CC will be derived as follows:

<table>
<thead>
<tr>
<th>If YY is:</th>
<th>CC becomes:</th>
</tr>
</thead>
<tbody>
<tr>
<td>69-99</td>
<td>19</td>
</tr>
<tr>
<td>00-38</td>
<td>20</td>
</tr>
<tr>
<td>39-68</td>
<td>ERROR</td>
</tr>
</tbody>
</table>

The resulting time will be affected by the value of the TZ environment variable. If the resulting time value precedes the Epoch, touch will exit immediately with an error status. The range of valid times is the Epoch to January 18, 2038.

The range for SS is [00-61] rather than [00-59] because of leap seconds. If SS is 60 or 61, and the resulting time, as affected by the TZ environment variable, does not refer to a leap second, the resulting time will be one or two seconds after a time where SS is 59. If SS is not given, it is assumed to be 0.

The following option is supported for the settime utility:

- \texttt{-f ref\_file} Uses the corresponding times of the file named by \texttt{ref\_file} instead of the current time.

OPERANDS

The following operands are supported for the touch and settime utilities:

\texttt{file} A path name of a file whose times are to be modified.

\texttt{date\_time} Uses the specified \texttt{date\_time} instead of the current time. This operand is a decimal number of the form:

\texttt{MM\_DDhh\_mm YY} where each two digits represent the following:

\begin{itemize}
  \item \texttt{MM} The month of the year [01-12].
  \item \texttt{DD} The day of the month [01-31].
  \item \texttt{hh} The hour of the day [00-23].
  \item \texttt{mm} The minute of the hour [00-59].
  \item \texttt{YY} The second two digits of the year.
\end{itemize}
YY is optional. If it is omitted, the current year will be assumed. If YY is specified, the year will be derived as follows:

<table>
<thead>
<tr>
<th>YY</th>
<th>Corresponding Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>00-38</td>
<td>2000-2038</td>
</tr>
<tr>
<td>39-68</td>
<td>ERROR</td>
</tr>
</tbody>
</table>

If no -r option is specified, no -t option is specified, at least two operands are specified, and the first operand is an eight- or ten-digit decimal integer, the first operand will be assumed to be a date_time operand; otherwise, the first operand will be assumed to be a file operand.

**USAGE**

See largefile(5) for the description of the behavior of `touch` when encountering files greater than or equal to 2 Gbyte (\(2^{31}\) bytes).

**ENVIRONMENT VARIABLES**

See environ(5) for descriptions of the following environment variables that affect the execution of `touch`: LANG, LC_ALL, LC_CTYPE, LC_MESSAGES, and NLSPATH.

- **TZ**: Determine the timezone to be used for interpreting the time option-argument or the date_time operand.

**EXIT STATUS**

The following exit values are returned:

- **0**: The `touch` utility executed successfully and all requested changes were made.
- **>0**: An error occurred. The `touch` utility returned the number of files for which the times could not be successfully modified.

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

**SEE ALSO**

time(2), attributes(5), environ(5), largefile(5)

**NOTES**

Users familiar with the BSD environment will find that for the `touch` utility, the `-f` option is accepted but ignored. The `-f` option is unnecessary because `touch` will succeed for all files owned by the user regardless of the permissions on the files.
touch(1B)

NAME  | touch – change file access and modification times
SYNOPSIS | /usr/ucb/touch [-acfm] file...
DESCRIPTION | touch sets the access and modification times of each file to the current time. file is created if it does not already exist.
OPTIONS | -a Change the access time of file. Do not change the modification time unless -m is also specified.
          -c Do not create file if it does not exist.
          -f Attempt to force the touch in spite of read and write permissions on file.
          -m Change the modification time of file. Do not change the access time unless -a is also specified.
USAGE | See largefile(5) for the description of the behavior of touch when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).
EXIT STATUS | The following exit values are returned:
0 | touch executed successfully and all requested changes were made.
>0 | An error occurred. touch returns the number of files for which the times could not be successfully modified.
ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>
SEE ALSO | touch(1), attributes(5), largefile(5)
t plot, t300, t300s, t4014, t450, tek, ver – graphics filters for various plotters

/usr/bin/tplot

![Tplot](https://via.placeholder.com/150)

**SYNOPSIS**

```
/usr/bin/tplot [-T terminal]
```

**DESCRIPTION**

`tplot` reads plotting instructions from the standard input and produces plotting instructions suitable for a particular `terminal` on the standard output.

If no `terminal` is specified, the environment variable `TERM` is used. The default `terminal` is `tek`.

**ENVIRONMENT VARIABLES**

Except for `ver`, the following terminal-types can be used with `’lpr -g’` (see `lpr`) to produce plotted output:

- 300: DASI 300 or GSI terminal (Diablo® mechanism).
- 300s | 300S: DASI 300s terminal (Diablo mechanism).
- 450: DASI Hyterm 450 terminal (Diablo mechanism).
- 4014 | tek: Tektronix 4014 and 4015 storage scope with Enhanced Graphics Module. (Use 4013 for Tektronix 4014 or 4015 without the Enhanced Graphics Module).
- ver: Versatec® D1200A printer-plotter. The output is scan-converted and suitable input to `’lpr -v’`.

**FILES**

```
/usr/lib/t300
/usr/lib/t300s
/usr/lib/t4014
/usr/lib/t450
/usr/lib/tek
/usr/lib/vplot
```

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

**SEE ALSO**

`lp(1), vi(1), attributes(5)"
tput(1)

NAME
tput – initialize a terminal or query terminfo database

SYNOPSIS
tput [-T type] capname [parm...]
tput -S <<

description
tput uses the terminfo database to make the values of terminal-dependent
capabilities and information available to the shell (see sh(1)); to clear, initialize or reset
the terminal; or to return the long name of the requested terminal type. tput outputs
a string if the capability attribute (capname) is of type string, or an integer if the
attribute is of type integer. If the attribute is of type boolean, tput simply sets the exit
status (0 for TRUE if the terminal has the capability, 1 for FALSE if it does not), and
produces no output. Before using a value returned on standard output, the user
should test the exit status ($?, see sh(1)) to be sure it is 0. See the EXIT STATUS
section.

options
The following options are supported:

-T type
Indicates the type of terminal. Normally this option is
unnecessary, because the default is taken from the environment
variable TERM. If -T is specified, then the shell variables LINES
and COLUMNS and the layer size will not be referenced.

-S
Allows more than one capability per invocation of tput. The
capabilities must be passed to tput from the standard input
instead of from the command line (see the example in the
EXAMPLES section). Only one capname is allowed per line. The -S
option changes the meaning of the 0 and 1 boolean and string exit
statuses (see the EXIT STATUS section).

operands
The following operands are supported:

capname
Indicates the capability attribute from the terminfo database. See
terminfo(4) for a complete list of capabilities and the capname
associated with each.

The following strings will be supported as operands by the
implementation in the "C" locale:

clear
Display the clear-screen sequence.

init
If the terminfo database is present and an
entry for the user's terminal exists (see -T type,
above), the following will occur:

1. if present, the terminal's initialization
   strings will be output (is1, is2, is3, if, iprog),
2. any delays (for instance, newline) specified
   in the entry will be set in the tty driver,
3. tabs expansion will be turned on or off according to the specification in the entry, and
4. if tabs are not expanded, standard tabs will be set (every 8 spaces). If an entry does not contain the information needed for any of the four above activities, that activity will silently be skipped.

reset
Instead of putting out initialization strings, the terminal’s reset strings will be output if present (rs1, rs2, rs3, rf). If the reset strings are not present, but initialization strings are, the initialization strings will be output. Otherwise, reset acts identically to init.

longname
If the terminfo database is present and an entry for the user’s terminal exists (see -T type above), then the long name of the terminal will be put out. The long name is the last name in the first line of the terminal’s description in the terminfo database (see term(5)).

dparm
If the attribute is a string that takes parameters, the argument parm will be instantiated into the string. An all numeric argument will be passed to the attribute as a number.

EXAMPLES

EXAMPLE 1 Using the tput command.

This example initializes the terminal according to the type of terminal in the environment variable TERM. This command should be included in everyone’s .profile after the environment variable TERM has been exported, as illustrated on the profile(4) manual page.

tput init
The next example resets an AT&T 5620 terminal, overriding the type of terminal in the environment variable TERM:

tput -T5620 reset
The following example sends the sequence to move the cursor to row 0, column 0 (the upper left corner of the screen, usually known as the "home" cursor position).

cup 0 0
The next example echos the clear-screen sequence for the current terminal.

clear
The next command prints the number of columns for the current terminal.

cols
The following command prints the number of columns for the 450 terminal.
EXAMPLE 1 Using the `tput` command.  (Continued)

```bash
example% tput -T450 cols
```

The next example sets the shell variables `bold`, to begin stand-out mode sequence,
and `offbold`, to end standout mode sequence, for the current terminal. This might be
followed by a prompt:

```bash
echo "${bold}Please type in your name: ${offbold}\c"
```

```bash
example% bold='tput smso'
example% offbold='tput rmso'
```

This example sets the exit status to indicate if the current terminal is a hardcopy
terminal.

```bash
example% tput hc
```

This next example sends the sequence to move the cursor to row 23, column 4.

```bash
example% tput cup 23 4
```

The next command prints the long name from the `terminfo` database for the type of
terminal specified in the environment variable `TERM`.

```bash
example% tput longname
```

This last example shows `tput` processing several capabilities in one invocation. This
element clears the screen, moves the cursor to position 10, 10 and turns on bold (extra
bright) mode. The list is terminated by an exclamation mark (!) on a line by itself.

```bash
example% tput -S <<!
  clear
  cup 10 10
  bold
  
```

See `environ(5)` for descriptions of the following environment variables that affect the
execution of `tput`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

**ENVIRONMENT VARIABLES**

- **TERM** Determine the terminal type. If this variable is unset or null, and if the `-T`
option is not specified, an unspecified default terminal type will be used.

**EXIT STATUS**

The following exit values are returned:

0

- If `capname` is of type boolean and `-S` is not specified, indicates `TRUE`.
- If `capname` is of type string and `-S` is not specified, indicates `capname` is
defined for this terminal type.
- If `capname` is of type boolean or string and `-S` is specified, indicates that
  all lines were successful.
- `capname` is of type integer.
- The requested string was written successfully.

1

- If `capname` is of type boolean and `-S` is not specified, indicates `FALSE`. 

1562 man pages section 1: User Commands • Last Revised 1 Feb 1995
If `capname` is of type string and `-S` is not specified, indicates that `capname` is not defined for this terminal type.

1. Usage error.
2. No information is available about the specified terminal type.
3. The specified operand is invalid.
4. An error occurred.

- `capname` is a numeric variable that is not specified in the terminfo database; for instance, `tput -T450 lines` and `tput -T2621 xmc`.

**FILES**

- `/usr/include/curses.h` curses(3CURSES) header
- `/usr/include/term.h` terminfo header
- `/usr/lib/tabset/*` tab settings for some terminals, in a format appropriate to be output to the terminal (escape sequences that set margins and tabs); for more information, see the "Tabs and Initialization" section of terminfo(4)
- `/usr/share/lib/terminfo/?/*` compiled terminal description database

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

**SEE ALSO**
clear(1), sh(1), stty(1), tabs(1), curses(3CURSES), profile(4), terminfo(4), attributes(5), environ(5), term(5)
The `tr` utility copies the standard input to the standard output with substitution or deletion of selected characters. The options specified and the `string1` and `string2` operands control translations that occur while copying characters and single-character collating elements.

### OPTIONS

The following options are supported:

- `-c` Complements the set of characters specified by `string1`.  
- `-d` Deletes all occurrences of input characters that are specified by `string1`.  
- `-s` Replaces instances of repeated characters with a single character.

When the `-d` option is not specified:

- Each input character found in the array specified by `string1` is replaced by the character in the same relative position in the array specified by `string2`. When the array specified by `string2` is shorter than the one specified by `string1`, the results are unspecified.
- If the `-c` option is specified, the complements of the characters specified by `string1` (the set of all characters in the current character set, as defined by the current setting of `LC_CTYPE`, except for those actually specified in the `string1` operand) are placed in the array in ascending collation sequence, as defined by the current setting of `LC_COLLATE`.
- Because the order in which characters specified by character class expressions or equivalence class expressions is undefined, such expressions should only be used if the intent is to map several characters into one. An exception is case conversion, as described previously.

When the `-d` option is specified:

- Input characters found in the array specified by `string1` will be deleted.
- When the `-c` option is specified with `-d`, all characters except those specified by `string1` will be deleted. The contents of `string2` will be ignored, unless the `-s` option is also specified.
- The same string cannot be used for both the `-d` and the `-s` option; when both options are specified, both `string1` (used for deletion) and `string2` (used for squeezing) are required.
When the -s option is specified, after any deletions or translations have taken place, repeated sequences of the same character will be replaced by one occurrence of the same character, if the character is found in the array specified by the last operand. If the last operand contains a character class, such as the following example:

```
tr -s '[:space:]'
```

the last operand’s array will contain all of the characters in that character class. However, in a case conversion, as described previously, such as

```
tr -s '[:upper:]' '[:lower:]
```

the last operand’s array will contain only those characters defined as the second characters in each of the toupper or tolower character pairs, as appropriate. (See toupper(3C) and tolower(3C)).

An empty string used for string1 or string2 produces undefined results.

**OPERANDS**

The following operands are supported:

- **string1**
  Translation control strings. Each string represents a set of characters to be converted into an array of characters used for the translation.

- **string2** (if specified) defines two arrays of characters. The constructs in the following list can be used to specify characters or single-character collating elements. If any of the constructs result in multi-character collating elements, tr will exclude, without a diagnostic, those multi-character elements from the resulting array.

- **character**
  Any character not described by one of the conventions below represents itself.

- **\ octal**
  Octal sequences can be used to represent characters with specific coded values. An octal sequence consists of a backslash followed by the longest sequence of one-, two-, or three-octal-digit characters (01234567). The sequence causes the character whose encoding is represented by the one-, two- or three-digit octal integer to be placed into the array. Multi-byte characters require multiple, concatenated escape sequences of this type, including the leading \ for each byte.

- **\ character**
  The backslash-escape sequences \a, \b, \f, \n, \r, \t, and \v are supported. The results of using any other character, other than an octal digit, following the backslash are unspecified.
c-c

Represented the range of collating elements between the range endpoints, inclusive, as defined by the current setting of the LC_COLLATE locale category. The starting endpoint must precede the second endpoint in the current collation order. The characters or collating elements in the range are placed in the array in ascending collation sequence.

[c-c]

Represents all characters belonging to the defined character class, as defined by the current setting of the LC_CTYPE locale category. The following character class names will be accepted when specified in string1:

alnum blank digit lower punct upper
alpha cntrl graph print space xdigit

In addition, character class expressions of the form [:name:] are recognized in those locales where the name keyword has been given a charclass definition in the LC_CTYPE category.

Note: /usr/bin/tr supports character class expressions only in singlebyte locales. Use /usr/xpg4/bin/tr to support these expressions in any locale.

When both the -d and -s options are specified, any of the character class names will be accepted in string2. Otherwise, only character class names lower or upper are valid in string2 and then only if the corresponding character class upper and lower, respectively, is specified in the same relative position in string1. Such a specification is interpreted as a request for case conversion. When [:lower:] appears in string1 and [:upper:] appears in string2, the arrays will contain the characters from the toupper mapping in the LC_CTYPE category of the current locale. When [:upper:] appears in string1 and [:lower:] appears in string2, the arrays will contain the characters from the tolower mapping in the LC_CTYPE category of the current locale. The first character from each mapping pair will be in the array for string1 and the second character from each mapping pair will be in the array for string2 in the same relative position.

Except for case conversion, the characters specified by a character class expression are placed in the array in an unspecified order.

If the name specified for class does not define a valid character class in the current locale, the behavior is undefined.

[:equiv:]

Represents all characters or collating elements belonging to the same equivalence class as equiv, as defined by the current setting of the LC_COLLATE locale category. An equivalence class expression
is allowed only in \textit{string1}, or in \textit{string2} when it is being used by the combined \texttt{-d} and \texttt{-s} options. The characters belonging to the equivalence class are placed in the array in an unspecified order.

\[
x^n
\]

Represents \( n \) repeated occurrences of the character \( x \). Because this expression is used to map multiple characters to one, it is only valid when it occurs in \textit{string2}. If \( n \) is omitted or is 0, it is interpreted as large enough to extend the \textit{string2}-based sequence to the length of the \textit{string1}-based sequence. If \( n \) has a leading 0, it is interpreted as an octal value. Otherwise, it is interpreted as a decimal value.

**Usage**

See \texttt{largefile(5)} for the description of the behavior of \texttt{tr} when encountering files greater than or equal to 2 Gbyte (\( 2^{31} \) bytes).

**Examples**

**Example 1** Creating a list of words

The following example creates a list of all words in \texttt{file1}, one per line in \texttt{file2}, where a word is taken to be a maximal string of letters.

\begin{verbatim}
tr -cs "[:alpha:]" "[\n*]" <file1 >file2
\end{verbatim}

**Example 2** Translating characters

This example translates all lower-case characters in \texttt{file1} to upper-case and writes the results to standard output.

\begin{verbatim}
tr "[:lower:]" "[:upper:]" <file1
\end{verbatim}

Notice that the caveat expressed in the corresponding example in XPG3 is no longer in effect. This case conversion is now a special case that employs the \texttt{tolower} and \texttt{toupper} classifications, ensuring that proper mapping is accomplished (when the locale is correctly defined).

**Example 3** Identifying equivalent characters

This example uses an equivalence class to identify accented variants of the base character \( e \) in \texttt{file1}, which are stripped of diacritical marks and written to \texttt{file2}.

\begin{verbatim}
tr "[=e=]" e <file1 >file2
\end{verbatim}

See \texttt{environ(5)} for descriptions of the following environment variables that affect the execution of \texttt{tr}: \texttt{LC_COLLATE}, \texttt{LC_CTYPE}, \texttt{LC_MESSAGES}, and \texttt{NLSPATH}.

**Exit Status**

The following exit values are returned:

<table>
<thead>
<tr>
<th>Exit Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>All input was processed successfully.</td>
</tr>
<tr>
<td>&gt;0</td>
<td>An error occurred.</td>
</tr>
</tbody>
</table>
See attributes(5) for descriptions of the following attributes:

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>Not enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWxcu4</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

SEE ALSO ed(1), sed(1), sh(1), tolower(3C), toupper(3C), ascii(5), attributes(5), environ(5), largefile(5), XPG4(5)

NOTES Unlike some previous versions, /usr/xpg4/bin/tr correctly processes NUL characters in its input stream. NUL characters can be stripped by using tr -d '\000'.
NAME

tr – translate characters

SYNOPSIS

/usr/ucb/tr [-cds] [string1 [string2]]

DESCRIPTION

The tr utility copies the standard input to the standard output with substitution or
deletion of selected characters. The arguments string1 and string2 are considered sets
of characters. Any input character found in string1 is mapped into the character in the
 corresponding position within string2. When string2 is short, it is padded to the length
of string1 by duplicating its last character.

In either string the notation:

a–b

denotes a range of characters from a to b in increasing ASCII order. The character \,
followed by 1, 2 or 3 octal digits stands for the character whose ASCII code is given by
those digits. As with the shell, the escape character \, followed by any other
character, escapes any special meaning for that character.

OPTIONS

Any combination of the options -c, -d, or -s may be used:

- c  Complement the set of characters in string1 with respect to the universe of
      characters whose ASCII codes are 01 through 0377 octal.
- d  Delete all input characters in string1.
- s  Squeeze all strings of repeated output characters that are in string2 to single
      characters.

EXAMPLES

EXAMPLE 1 Creating a list of all the words in a filename

The following example creates a list of all the words in filename1, one per line, in
filename2, where a word is taken to be a maximal string of alphabetics. The second
string is quoted to protect ‘ \ ’ from the shell. 012 is the ASCII code for NEWLINE.

example1 tr -cs A-Za-z ‘\012’ < filename1 > filename2

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

SEE ALSO

ed(1), ascii(5), attributes(5)

NOTES

Will not handle ASCII NUL in string1 or string2. tr always deletes NUL from input.
trap(1)

NAME     trap, onintr – shell built-in functions to respond to (hardware) signals

sh        trap [argument n [n2...]]

csh       onintr [- | label]

ksh       *trap [arg sig [sig2...]]

sh        The trap command argument is to be read and executed when the shell receives
numeric or symbolic signal(s) (n). (Note: argument is scanned once when the trap is set
and once when the trap is taken.) Trap commands are executed in order of signal
number or corresponding symbolic names. Any attempt to set a trap on a signal that
was ignored on entry to the current shell is ineffective. An attempt to trap on signal 11
(memory fault) produces an error. If argument is absent all trap(s) n are reset to their
original values. If argument is the null string this signal is ignored by the shell and by
the commands it invokes. If n is 0 the command argument is executed on exit from the
shell. The trap command with no arguments prints a list of commands associated
with each signal number.

csh       onintr controls the action of the shell on interrupts. With no arguments, onintr
restores the default action of the shell on interrupts. (The shell terminates shell scripts
and returns to the terminal command input level). With the – argument, the shell
ignores all interrupts. With a label argument, the shell executes a goto label when an
interrupt is received or a child process terminates because it was interrupted.

ksh       trap uses arg as a command to be read and executed when the shell receives signal(s)
sig. (Note that arg is scanned once when the trap is set and once when the trap is
taken.) Each sig can be given as a number or as the name of the signal. trap
commands are executed in order of signal number. Any attempt to set a trap on a
signal that was ignored on entry to the current shell is ineffective. If arg is omitted or is
–, then the trap(s) for each sig are reset to their original values. If arg is the null (the
empty string, e.g., "") string then this signal is ignored by the shell and by the
commands it invokes. If sig is ERR then arg will be executed whenever a command has
a non-zero exit status. If sig is DEBUG then arg will be executed after each command. If
sig is 0 or EXIT for a trap set outside any function then the command arg is executed
on exit from the shell. The trap command with no arguments prints a list of
commands associated with each signal number.

On this man page, ksh(1) commands that are preceded by one or two * (asterisks) are
treated specially in the following ways:

1. Variable assignment lists preceding the command remain in effect when the
   command completes.

2. I/O redirections are processed after variable assignments.

3. Errors cause a script that contains them to abort.

4. Words, following a command preceded by ** that are in the format of a variable
   assignment, are expanded with the same rules as a variable assignment. This
   means that tilde substitution is performed after the = sign and word splitting and
   file name generation are not performed.
See attributes(5) for descriptions of the following attributes:

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO csh(1), exit(1), ksh(1), sh(1), attributes(5)
troff(1)

NAME
troff – typeset or format documents

SYNOPSIS

[-Tdest] [-uN] [-z] [filename...]

DESCRIPTION
troff formats text in the filenames for typesetting or laser printing. Input to troff is expected to consist of text interspersed with formatting requests and macros. If no filename argument is present, troff reads standard input. A minus sign (−) as a filename indicates that standard input should be read at that point in the list of input files.

The output of troff is usually piped through dpost(1) to create a printable postscript file (see EXAMPLES).

OPTIONS

The following options are supported. They may appear in any order, but all must appear before the first filename.

-a Send an ASCII approximation of formatted output to standard output. (Note: a rough ASCII version can also be printed out on ordinary terminals with an old and rarely used command, /usr/bin/ta.)

-f Do not print a trailer after the final page of output or cause the postprocessor to relinquish control of the device.

-Fdir Search directory dir for font width or terminal tables instead of the system default directory.

-i Read standard input after all input files are exhausted.

-m name Prepend the macro file /usr/share/lib/tmac/name to the input filenames. Note: most references to macro packages include the leading m as part of the name; for example, the man(5) macros reside in /usr/share/lib/tmac/an. The macro directory can be changed by setting the TROFFMACS environment variable to a specific path. Be certain to include the trailing ‘ ’ (slash) at the end of the path.

-nN Number the first generated page N.

-olist Print only pages whose page numbers appear in the comma-separated list of numbers and ranges. A range N–M means pages N through M; an initial −N means from the beginning to page N; and a final N– means from N to the end.

-q Quiet mode in nroff; ignored in troff.

-raN Set register a (one-character names only) to N.

-sN Stop the phototypesetter every N pages. On some devices, troff produces a trailer so you can change cassettes; resume by pressing the typesetter’s start button.
troff(1)

-Tdest Prepare output for typesetter dest. The following values can be supplied for dest:

post A PostScript printer; this is the default value. The output of the -T option must go through dpost(1) before it is sent to a PostScript printer to obtain the proper output.

aps Autologic APS-5.

-uN Set the emboldening factor for the font mounted in position 3 to N. If N is missing, then set the emboldening factor to 0.

-z Suppress formatted output. Only diagnostic messages and messages output using the .tm request are output.

OPERANDS The following operand is supported:

filename The file containing text to be processed by troff.

EXAMPLES EXAMPLE 1 Using troff

The following example shows how to print an input file mytext, coded with formatting requests and macros. The input file contains equations and tables and must go through the tbl(1) and eqn(1) preprocessors before it is formatted by troff with ms macros, processed by dpost(1), and printed by lp(1):

tbl mytext | eqn | troff -ms | dpost | lp

FILES /tmp/trtmp temporary file
/usr/share/lib/tmac/* standard macro files
/usr/lib/font/* font width tables for alternate mounted troff fonts
/usr/share/lib/nterm/* terminal driving tables for nroff

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWdoc</td>
</tr>
</tbody>
</table>

SEE ALSO checknr(1), col(1), dpost(1), eqn(1), lp(1), man(1), nroff(1), tbl(1), attributes(5), man(5), me(5), ms(5)

NOTES troff is not 8-bit clean because it is by design based on 7-bit ASCII.
Previous documentation incorrectly described the numeric register \texttt{yr} as being the "Last two digits of current year". \texttt{yr} is in actuality the number of years since 1900. To correctly obtain the last two digits of the current year through the year 2099, the definition given below of string register \texttt{yy} may be included in a document and subsequently used to display a two-digit year. Note that any other available one- or two-character register name may be substituted for \texttt{yy}.

\begin{verbatim}
." definition of new string register yy--last two digits of year
." use yr (# of years since 1900) if it is < 100
.ie \n{yr<100 .ds yy \n}yr
.el \{ ." else, subtract 100 from yr, store in ny
.nr ny \n{yr-100
.ie \n{ny<10 \{
}" use ny if it is two digits
.ds yy \n{ny
." remove temporary number register ny
.rr ny \}
.el \{.ds yy 0
." if ny is one digit, append it to 0
.as yy \n{ny
.rr ny \} \}
\end{verbatim}
true, false – provide truth values

SYNOPSIS

true
false

DESCRIPTION

ttrue does nothing, successfully. false does nothing, unsuccessfully. They are typically used in a shell script sh as:

```
while true
do
  command
done
```

which executes command forever.

EXIT STATUS

ttrue has exit status 0.
false always will exit with a non-zero value.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO

sh(1), attributes(5)
The `truss` utility executes the specified command and produces a trace of the system calls it performs, the signals it receives, and the machine faults it incurs. Each line of the trace output reports either the fault or signal name or the system call name with its arguments and return value(s). System call arguments are displayed symbolically when possible using defines from relevant system headers; for any path name pointer argument, the pointed-to string is displayed. Error returns are reported using the error code names described in `intro(3)`.

Optionally (see the `-u` option), `truss` will also produce an entry/exit trace of user-level function calls executed by the traced process, indented to indicate nesting.

The following options are recognized. For those options that take a list argument, the name `all` can be used as a shorthand to specify all possible members of the list. If the list begins with a `!`, the meaning of the option is negated (for example, exclude rather than trace). Multiple occurrences of the same option may be specified. For the same name in a list, subsequent options (those to the right) override previous ones (those to the left).

```
$p  Interprets the command arguments to `truss` as a list of process-ids for existing processes (see `ps(1)` rather than as a command to be executed. `truss` takes control of each process and begins tracing it provided that the userid and groupid of the process match those of the user or that the user is a privileged user. Processes may also be specified by their names in the `/proc` directory, for example, `/proc/12345`.
```

```
-f  Follows all children created by `fork()` or `vfork()` and includes their signals, faults, and system calls in the trace output. Normally, only the first-level command or process is traced. When `-f` is specified, the process-id is included with each line of trace output to indicate which process executed the system call or received the signal.
```

```
-c  Counts traced system calls, faults, and signals rather than displaying the trace line-by-line. A summary report is produced after the traced command terminates or when `truss` is interrupted. If `-f` is also specified, the counts include all traced system calls, faults, and signals for child processes.
```

```
-a  Shows the argument strings that are passed in each `exec()` system call.
```

```
-e  Shows the environment strings that are passed in each `exec()` system call.
```
Do not display interruptible sleeping system calls. Certain system calls, such as `open()` and `read()` on terminal devices or pipes, can sleep for indefinite periods and are interruptible. Normally, `truss` reports such sleeping system calls if they remain asleep for more than one second. The system call is reported again a second time when it completes. The `-i` option causes such system calls to be reported only once, when they complete.

-`l`  
Includes the id of the responsible lightweight process (LWP) with each line of trace output. If `-f` is also specified, both the process-id and the LWP-id are included.

-`d`  
Includes a time stamp on each line of trace output. The time stamp appears as a field containing `seconds . fraction` at the start of the line. This represents a time in seconds relative to the beginning of the trace. The first line of the trace output will show the base time from which the individual time stamps are measured, both as seconds since the epoch (see `time(2)`) and as a date string (see `ctime(3C)` and `date(1)`). The times that are reported are the times that the event in question occurred. For all system calls, the event is the completion of the system call, not the start of the system call.

-`D`  
Includes a time delta on each line of trace output. The value appears as a field containing `seconds . fraction` and represents the elapsed time for the LWP that incurred the event since the last reported event incurred by that LWP. Specifically, for system calls, this is not the time spent within the system call.

-`t` `[!]syscall, ...`  
System calls to trace or exclude. Those system calls specified in the comma-separated list are traced. If the list begins with a `![` , the specified system calls are excluded from the trace output. Default is `-tall`.

-`T` `[!]syscall, ...`  
System calls that stop the process. The specified system calls are added to the set specified by `-t`. If one of the specified system calls is encountered, `truss` leaves the process stopped and abandoned. That is, `truss` releases the process and exits but leaves the process in the stopped state at completion of the system call in question. A debugger or other process inspection tool (see `proc(1)` ) can then be applied to the stopped process. `truss` can be reapplied to the stopped process with the same or different options to continue tracing. Default is `-T!all`.

A process left stopped in this manner cannot be restarted by the application of `kill -CONT` because it is stopped on an event of interest via `/proc`, not by the default action of a stopping signal (see `signal(3HEAD)`). The `prun(1)` command described in `proc(1)` can be used to set the stopped process running again.

-`v` `[!]syscall, ...`  
Verbose. Displays the contents of any structures passed by address to the specified system calls (if traced by `-t`). Input values as well as values returned by the
operating system are shown. For any field used as both input and output, only the output value is shown. Default is \texttt{-v!all}.

\texttt{-x [!]syscall, \ldots}
Displays the arguments to the specified system calls (if traced by \texttt{-t}) in raw form, usually hexadecimal, rather than symbolically. This is for unredeemed hackers who must see the raw bits to be happy. Default is \texttt{-x!all}.

\texttt{-s [!]signal, \ldots}
Signals to trace or exclude. Those signals specified in the comma-separated list are traced. The trace output reports the receipt of each specified signal, even if the signal is being ignored (not blocked). (Blocked signals are not received until they are unblocked.) Signals may be specified by name or number (see \texttt{<sys/signal.h>}). If the list begins with a \texttt{!}, the specified signals are excluded from the trace output. Default is \texttt{-s!all}.

\texttt{-S [!]signal, \ldots}
Signals that stop the process. The specified signals are added to the set specified by \texttt{-s}. If one of the specified signals is received, \texttt{truss} leaves the process stopped and abandoned (see the \texttt{-T} option). Default is \texttt{-S!all}.

\texttt{-m [!]fault, \ldots}
Machine faults to trace or exclude. Those faults specified in the comma-separated list are traced. Faults may be specified by name or number (see \texttt{<sys/fault.h>}). If the list begins with a \texttt{!}, the specified faults are excluded from the trace output. Default is \texttt{-m!all}.

\texttt{-M [!]fault, \ldots}
Machine faults that stop the process. The specified faults are added to the set specified by \texttt{-m}. If one of the specified faults is incurred, \texttt{truss} leaves the process stopped and abandoned (see the \texttt{-T} option). Default is \texttt{-M!all}.

\texttt{-r [!]fd, \ldots}
Shows the full contents of the I/O buffer for each \texttt{read()} on any of the specified file descriptors. The output is formatted 32 bytes per line and shows each byte as an ASCII character (preceded by one blank) or as a 2-character C language escape sequence for control characters such as horizontal tab (\texttt{\t}) and newline (\texttt{\n}). If ASCII interpretation is not possible, the byte is shown in 2-character hexadecimal representation. (The first 12 bytes of the I/O buffer for each traced \texttt{read()} are shown even in the absence of \texttt{-r}.) Default is \texttt{-r!all}.

\texttt{-w [!]fd, \ldots}
Shows the contents of the I/O buffer for each \texttt{write()} on any of the specified file descriptors (see the \texttt{-r} option). Default is \texttt{-w!all}.

\texttt{-u [!]lib, \ldots[:][!]func, \ldots}
User-level function call tracing. \texttt{lib, \ldots} is a comma-separated list of dynamic library names, excluding the “.so.n” suffix. \texttt{func, \ldots} is a comma-separated list of function names. In both cases the names can include name-matching metacharacters *, ?, [ ] with the same meanings as those of \texttt{sh(1)} but as applied to the library/function name spaces, not to files. An empty library or function list defaults to *, trace all libraries or functions in a library. A leading \texttt{!} on either list specifies an exclusion
list, names of libraries or functions not to be traced. Excluding a library excludes all
functions in that library; any function list following a library exclusion list is
ignored.

A single : separating the library list from the function list means to trace calls into
the libraries from outside the libraries, but omit calls made to functions in a library
from other functions in the same library. A double :: means to trace all calls,
regardless of origin.

Library patterns do not match either the executable file or the dynamic linker
unless there is an exact match (1* will not match ld.so.1). To trace functions in
either of these objects, the names must be specified exactly, as in: truss -u a.out
-u ld ... a.out is the literal name to be used for this purpose; it does not stand
for the name of the executable file. Tracing a.out function calls implies all calls
(default is : :).

Multiple -u options may be specified and they are honored left-to-right. If the
process is linked with -lthread, the id of the thread that performed the function
call is included in the trace output for the call. truss searches the dynamic symbol
table in each library to find function names and will also search the standard
symbol table if it has not been stripped.

-u [!]lib, ... ;[!]func, ...
User-level function calls that stop the process. The specified functions are added to
the set specified by -u. If one of the specified functions is called, truss leaves the
process stopped and abandoned (see the -T option).

-o outfile
File to be used for the trace output. By default, the output goes to standard error.

See man pages section 2: System Calls for system call names accepted by the -t, -T, -v,
and -x options. System call numbers are also accepted.

If truss is used to initiate and trace a specified command and if the -o option is used
or if standard error is redirected to a non-terminal file, then truss runs with hangup,
interrupt, and quit signals ignored. This facilitates tracing of interactive programs that
catch interrupt and quit signals from the terminal.

If the trace output remains directed to the terminal, or if existing processes are traced
(the -p option), then truss responds to hangup, interrupt, and quit signals by
releasing all traced processes and exiting. This enables the user to terminate excessive
trace output and to release previously-existing processes. Released processes continue
normally, as though they had never been touched.

EXAMPLES

EXAMPLE 1 Tracing a command

This example produces a trace of the find(1) command on the terminal:

example$ truss find . -print >find.out

User Commands    1579
EXAMPLE 1 Tracing a command (Continued)

EXAMPLE 2 Tracing common system calls
To see only a trace of the open, close, read, and write system calls:
```
example$ truss -t open,close,read,write find . -print >find.out
```

EXAMPLE 3 Tracing a shell script
This produces a trace of the `spell(1)` command on the file `truss.out`:
```
example$ truss -f -o truss.out spell document
```

spell is a shell script, so the `-f` flag is needed to trace not only the shell but also the processes created by the shell. (The spell script runs a pipeline of eight processes.)

EXAMPLE 4 Abbreviating output
A particularly boring example is:
```
example$ truss nroff -mm document >nroff.out
```
because 97% of the output reports `lseek()`, `read()`, and `write()` system calls. To abbreviate it:
```
example$ truss -t ! lseek,read,write nroff -mm document >nroff.out
```

EXAMPLE 5 Tracing library calls from outside the C library
This example traces all user-level calls made to any function in the C library from outside the C library:
```
example$ truss -u libc . . .
```

EXAMPLE 6 Tracing library calls from within the C library
This example includes calls made to functions in the C library from within the C library itself:
```
example$ truss -u libc : : . . .
```

EXAMPLE 7 Tracing library calls other than the C library
This example traces all user-level calls made to any library other than the C library:
```
example$ truss -u `*` -u !libc . . .
```

EXAMPLE 8 Tracing printf and scanf function calls
This example traces all user-level calls to functions in the printf and scanf family contained in the C library:
EXAMPLE 8  Tracing printf and scanf function calls  (Continued)

```
example$ truss -u 'libc : *printf,*scanf' ...
```

EXAMPLE 9  Tracing any user-level function call

This example traces every user-level function call from anywhere to anywhere:

```
example$ truss -u a.out -u ld : : -u : : ...
```

EXAMPLE 10  Tracing a system call verbosely

This example verbosely traces the system call activity of process #1, init(1M) (if you are a privileged user):

```
example$ truss -p -v all 1
```

Interrupting truss returns init to normal operation.

FILES  /proc/*           process files

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWtoo (32-bit)</td>
</tr>
<tr>
<td></td>
<td>SUNWtoox (64-bit)</td>
</tr>
</tbody>
</table>

SEE ALSO  date(1), find(1), proc(1), ps(1), sh(1), spell(1), init (1M), intro(3), exec(2), fork(2), lseek(2), open(2), read(2), time(2), vfork(2), write(2), ctime(3C), threads(3THR), proc(4), attributes(5), signal(3HEAD)

man pages section 2: System Calls

NOTES  Some of the system calls described in man pages section 2: System Calls differ from the actual operating system interfaces. Do not be surprised by minor deviations of the trace output from the descriptions in that document.

Every machine fault (except a page fault) results in the posting of a signal to the LWP that incurred the fault. A report of a received signal will immediately follow each report of a machine fault (except a page fault) unless that signal is being blocked.

The operating system enforces certain security restrictions on the tracing of processes. In particular, any command whose object file (a.out) cannot be read by a user cannot be traced by that user; set-uid and set-gid commands can be traced only by a privileged user. Unless it is run by a privileged user, truss loses control of any process that performs an exec() of a set-id or unreadable object file; such processes continue normally, though independently of truss, from the point of the exec().
To avoid collisions with other controlling processes, truss will not trace a process that it detects is being controlled by another process via the /proc interface. This allows truss to be applied to proc(4)-based debuggers as well as to another instance of itself.

The trace output contains tab characters under the assumption that standard tab stops are set (every eight positions).

The trace output for multiple processes or for a multithreaded process (one that contains more than one LWP) is not produced in strict time order. For example, a read() on a pipe may be reported before the corresponding write(). For any one LWP (a traditional process contains only one), the output is strictly time-ordered.

When tracing more than one process, truss runs as one controlling process for each process being traced. For the example of the spell command shown above, spell itself uses 9 process slots, one for the shell and 8 for the 8-member pipeline, while truss adds another 9 processes, for a total of 18.

Not all possible structures passed in all possible system calls are displayed under the -v option.
NAME

tset, reset – establish or restore terminal characteristics

SYNOPSIS

tset [-InQrs] [-ec] [-kc] [-m [port-ID [baudrate] : type...]] [type]
baudrate] : type...]] [type]

DESCRIPTION

The tset utility sets up your terminal, typically when you first log in. It does terminal dependent processing such as setting erase and kill characters, setting or resetting delays, sending any sequences needed to properly initialized the terminal, and the like. tset first determines the type of terminal involved, and then does necessary initializations and mode settings. If a port is not wired permanently to a specific terminal (not hardwired) it is given an appropriate generic identifier such as dialup.

reset clears the terminal settings by turning off CBREAK and RAW modes, output delays and parity checking, turns on NEWLINE translation, echo and TAB expansion, and restores undefined special characters to their default state. It then sets the modes as usual, based on the terminal type (which will probably override some of the above). See stty(1) for more information. All arguments to tset may be used with reset. reset also uses rs= and rf= to reset the initialization string and file. This is useful after a program dies and leaves the terminal in a funny state. Often in this situation, characters will not echo as you type them. You may have to type LINEFEED reset LINEFEED since RETURN may not work.

When no arguments are specified, tset reads the terminal type from the TERM environment variable and re-initializes the terminal, and performs initialization of mode, environment and other options at login time to determine the terminal type and set up terminal modes.

When used in a startup script (.profile for sh(1) users or .login for csh(1) users) it is desirable to give information about the type of terminal you will usually use on ports that are not hardwired. Any of the alternate generic names given in the file /etc/termcap are possible identifiers. Refer to the -m option below for more information. If no mapping applies and a final type option, not preceded by a -m, is given on the command line then that type is used.

It is usually desirable to return the terminal type, as finally determined by tset, and information about the terminal’s capabilities, to a shell’s environment. This can be done using the −, -s, or -S options.

For the Bourne shell, put this command in your .profile file:

```
eval 'tset -s options....'
```

or using the C shell, put these commands in your .login file:

```
set noglob
eval 'tset -s options....' unset noglob
```

With the C shell, it is also convenient to make an alias in your .cshrc file:

```
alias ts 'eval 'tset -s \!*''
```
This also allows the command:

```
tset 2621
```

to be invoked at any time to set the terminal and environment. It is not possible to get
this aliasing effect with a Bourne shell script, because shell scripts cannot set the
environment of their parent. If a process could set its parent’s environment, none of
this nonsense would be necessary in the first place.

Once the terminal type is known, `tset` sets the terminal driver mode. This normally
involves sending an initialization sequence to the terminal, setting the single character
erase (and optionally the line-kill (full line erase)) characters, and setting special
character delays. TAB and NEWLINE expansion are turned off during transmission of
the terminal initialization sequence.

On terminals that can backspace but not overstrike (such as a CRT), and when the
erase character is ‘#’, the erase character is changed as if `-e` had been used.

**OPTIONS**

- `-`  The name of the terminal finally decided upon is output on the
  standard output. This is intended to be captured by the shell and placed in the `TERM` environment variable.

- `-ec` Set the erase character to be the named character `c` on all terminals. Default is the BACKSPACE key on the keyboard, usually `^H` (CTRL-H). The character `c` can either be typed directly, or entered using the circumflex-character notation used here.

- `-ic` Set the interrupt character to be the named character `c` on all terminals. Default is `^C` (CTRL-C). The character `c` can either be typed directly, or entered using the circumflex-character notation used here.

- `-i` Suppress transmitting terminal-initialization strings.

- `-k`  Set the line kill character to be the named character `c` on all terminals. Default is `^U` (CTRL-U). The kill character is left alone if `-k` is not specified. Control characters can be specified by prefixing the alphabetical character with a circumflex (as in CTRL-U) instead of entering the actual control key itself. This allows you to specify control keys that are currently assigned.

- `-n`  Specify that the new tty driver modes should be initialized for this
  terminal. Probably useless since `stty new` is the default.

- `-Q` Suppress printing the ‘Erase set to’ and ‘Kill set to’
  messages.

- `-r` In addition to other actions, reports the terminal type.

- `-s` Output commands to set and export `TERM`. This can be used with

```
set noglob
eval ‘tset -s ...’
unset noglob
```
to bring the terminal information into the environment. Doing so makes programs such as vi(1) start up faster. If the SHELL environment variable ends with csh, C shell commands are output, otherwise Bourne shell commands are output.

```
-m [ port-ID [ baudrate ] : type ] ...
```

Specify (map) a terminal type when connected to a generic port (such as dialup or plugboard) identified by port-ID. The baudrate argument can be used to check the baudrate of the port and set the terminal type accordingly. The target rate is prefixed by any combination of the following operators to specify the conditions under which the mapping is made:

- `>` Greater than
- `@` Equals or “at”
- `<` Less than
- `!` It is not the case that (negates the above operators)
- `?` Prompt for the terminal type. If no response is given, then type is selected by default.

In the following example, the terminal type is set to adm3a if the port is a dialup with a speed of greater than 300 or to dw2 if the port is a dialup at 300 baud or less. In the third case, the question mark preceding the terminal type indicates that the user is to verify the type desired. A NULL response indicates that the named type is correct. Otherwise, the user’s response is taken to be the type desired.

```
tset -m 'dialup>300:adm3a' -m 'dialup:dw2' -m 'plugboard:?adm3a'
```

To prevent interpretation as metacharacters, the entire argument to `-m` should be enclosed in single quotes. When using the C shell, exclamation points should be preceded by a backslash (\).

### EXAMPLES

These examples all use the ‘−’ option. A typical use of `tset` in a `.profile` or `.login` will also use the `-e` and `-k` options, and often the `-n` or `-Q` options as well. These options have been omitted here to keep the examples short.

#### EXAMPLE 1 Selecting a terminal

To select a 2621, you might put the following sequence of commands in your `.login` file (or `.profile` for Bourne shell users).

```
set noglob
eval 'tset -s 2621' 
unset noglob
```

If you want to make the selection based only on the baud rate, you might use the following:

```
set noglob
eval 'tset -s -m '>1200:wy' 2621' 
unset noglob
```
EXAMPLE 1 Selecting a terminal  (Continued)

EXAMPLE 2 Selecting terminals according to speed or baud rate

If you have a switch which connects to various ports (making it impractical to identify which port you may be connected to), and use various terminals from time to time, you can select from among those terminals according to the speed or baud rate. In the example below, tset will prompt you for a terminal type if the baud rate is greater than 1200 (say, 9600 for a terminal connected by an RS-232 line), and use a Wyse® 50 by default. If the baud rate is less than or equal to 1200, it will select a 2621. Note the placement of the question mark, and the quotes to protect the > and ? from interpretation by the shell.

```bash
set noglob
eval 'tset -s -m "switch>1200:?wy -m "switch<1200:2621""
unset noglob
```

EXAMPLE 3 Selecting the terminal used most often

The following entry is appropriate if you always dial up, always at the same baud rate, on many different kinds of terminals, and the terminal you use most often is an adm3a.

```bash
set noglob
eval 'tset -s ?adm3a'
unset noglob
```

EXAMPLE 4 Selecting a terminal with specific settings

The following example quietly sets the erase character to BACKSPACE, and kill to CTRL-U. If the port is switched, it selects a Concept™ 100 for speeds less than or equal to 1200, and asks for the terminal type otherwise (the default in this case is a Wyse 50). If the port is a direct dialup, it selects Concept 100 as the terminal type. If logging in over the ARPANET, the terminal type selected is a Datamedia® 2500 terminal or emulator. Note the backslash escaping the NEWLINE at the end of the first line in the example.

```bash
set noglob
eval 'tset -e -k\''U -Q -s -m "switch<1200:concept100" -m \\
    "switch:?wy" -m dialup:concept100 -m arpanet:dm2500' 
unset noglob
```
EXAMPLE 4 Selecting a terminal with specific settings (Continued)

FILES

.login
.profile
/etc/termcap

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

SEE ALSO csh(1), sh(1), stty(1), vi(1), attributes(5), environ(5)

NOTES

The tset command is one of the first commands a user must master when getting started on a UNIX system. Unfortunately, it is one of the most complex, largely because of the extra effort the user must go through to get the environment of the login shell set. Something needs to be done to make all this simpler, either the login program should do this stuff, or a default shell alias should be made, or a way to set the environment of the parent should exist.

This program cannot intuit personal choices for erase, interrupt and line kill characters, so it leaves these set to the local system standards.

It could well be argued that the shell should be responsible for ensuring that the terminal remains in a sane state; this would eliminate the need for the reset program.
The `tsort` command produces on the standard output a totally ordered list of items consistent with a partial ordering of items mentioned in the input file.

The input consists of pairs of items (nonempty strings) separated by blanks. Pairs of different items indicate ordering. Pairs of identical items indicate presence, but not ordering.

The following operand is supported:

- **file**: A path name of a text file to order. If no `file` operand is given, the standard input is used.

**Examples**

**EXAMPLE 1** An example of the `tsort` command.

The command:

```bash
tsort <<EOF
  a b c c d e
g g 
f g e f
EOF
```

produces the output:

```
a b c d e f g
```

**Environment Variables**

See `environ(5)` for descriptions of the following environment variables that affect the execution of `tsort`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

**Exit Status**

The following exit values are returned:

- **0**: Successful completion.
- **>0**: An error occurred.

**Attributes**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWbtool</td>
</tr>
</tbody>
</table>

**See Also**

`lorder(1)`, `attributes(5)`, `environ(5)`
DIAGNOSTICS: Odd data: there are an odd number of fields in the input file.
tty(1)

NAME
tty – return user’s terminal name

SYNOPSIS
tty [-l] [-s]

DESCRIPTION
The tty utility writes to the standard output the name of the terminal that is open as standard input. The name that is used is equivalent to the string that would be returned by the ttyname(3C) function.

OPTIONS
The following options are supported:
- 1 Prints the synchronous line number to which the user’s terminal is connected, if it is on an active synchronous line.
- s Inhibits printing of the terminal path name, allowing one to test just the exit status.

ENVIRONMENT VARIABLES
See environ(5) for descriptions of the following environment variables that affect the execution of tty: LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS
The following exit values are returned:
0 Standard input is a terminal.
1 Standard input is not a terminal.
>1 An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

SEE ALSO
isatty(3C), ttyname(3C), attributes(5), environ(5)

DIAGNOSTICS

- not on an active synchronous line
  The standard input is not a synchronous terminal and -1 is specified.
- not a tty
  The standard input is not a terminal and -s is not specified.

NOTES
The -s option is useful only if the exit status is wanted. It does not rely on the ability to form a valid path name. Portable applications should use test -t.
NAME  type – write a description of command type
SYNOPSIS  type name...
DESCRIPTION  The type utility indicates how each name operand would be interpreted if used as a command. type displays information about each operand identifying the operand as a shell built-in, function, alias, hashed command, or keyword, and where applicable, may display the operand’s path name.

There is also a shell built-in version of type that is similar to the type utility.

OPERANDS  The following operand is supported:

name  A name to be interpreted.
ENVIRONMENT VARIABLES  See environ(5) for descriptions of the following environment variables that affect the execution of type: LC_CTYPE, LC_MESSAGES, and NLSPATH.

PATH  Determine the location of name.
EXIT STATUS  The following exit values are returned:

0  Successful completion.
>0  An error occurred.
ATTRIBUTES  See attributes(5) for descriptions of the following attributes:

+------------------------------------------+------------------+
<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO  typeset(1), attributes(5), environ(5)
NAME  typeset, whence - shell built-in functions to set/get attributes and values for shell variables and functions

SYNOPSIS  typeset [± HLRZfilrtux [n]] [name [= value]]...
whence [-pv] name...

DESCRIPTION  typeset sets attributes and values for shell variables and functions. When typeset is invoked inside a function, a new instance of the variables name is created. The variables value and type are restored when the function completes. The following list of attributes may be specified:

- H  This flag provides UNIX to host-name file mapping on non-UNIX machines.

- L  Left justify and remove leading blanks from value. If n is non-zero it defines the width of the field; otherwise, it is determined by the width of the value of first assignment. When the variable is assigned to, it is filled on the right with blanks or truncated, if necessary, to fit into the field. Leading zeros are removed if the -Z flag is also set. The -R flag is turned off.

- R  Right justify and fill with leading blanks. If n is non-zero it defines the width of the field, otherwise it is determined by the width of the value of first assignment. The field is left filled with blanks or truncated from the end if the variable is reassigned. The -L flag is turned off.

- Z  Right justify and fill with leading zeros if the first non-blank character is a digit and the -L flag has not been set. If n is non-zero it defines the width of the field; otherwise, it is determined by the width of the value of first assignment.

- f  The names refer to function names rather than variable names. No assignments can be made and the only other valid flags are -t, -u and -x. The flag -t turns on execution tracing for this function. The flag -u causes this function to be marked undefined. The FPATH variable will be searched to find the function definition when the function is referenced. The flag -x allows the function definition to remain in effect across shell procedures invoked by name.

- i  Parameter is an integer. This makes arithmetic faster. If n is non-zero it defines the output arithmetic base; otherwise, the first assignment determines the output base.

- l  All upper-case characters are converted to lower-case. The upper-case flag, -u is turned off.

- r  The given names are marked readonly and these names cannot be changed by subsequent assignment.

- t  Tags the variables. Tags are user definable and have no special meaning to the shell.
-u All lower-case characters are converted to upper-case characters. The lower-case flag, -l, is turned off.

-x The given names are marked for automatic export to the environment of subsequently-executed commands.

The -i attribute can not be specified along with -R, -L, -Z, or -f.

Using + rather than − causes these flags to be turned off. If no name arguments are given but flags are specified, a list of names (and optionally the values) of the variables which have these flags set is printed. (Using + rather than − keeps the values from being printed.) If no names and flags are given, the names and attributes of all variables are printed.

For each name, whence indicates how it would be interpreted if used as a command name.

The -v flag produces a more verbose report.

The -p flag does a path search for name even if name is an alias, a function, or a reserved word.

On this man page, ksh(1) commands that are preceded by one or two * (asterisks) are treated specially in the following ways:

1. Variable assignment lists preceding the command remain in effect when the command completes.
2. I/O redirections are processed after variable assignments.
3. Errors cause a script that contains them to abort.
4. Words, following a command preceded by ** that are in the format of a variable assignment, are expanded with the same rules as a variable assignment. This means that tilde substitution is performed after the = sign and word splitting and file name generation are not performed.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

**SEE ALSO** ksh(1), set(1), sh(1), attributes(5)
ucblinks(1B)

NAME  ucblinks – adds /dev entries to give SunOS 4.x compatible names to SunOS 5.x devices

SYNOPSIS  /usr/ucb/ucblinks [-e rulebase] [-r rootdir]

DESCRIPTION  ucblinks creates symbolic links under the /dev directory for devices whose SunOS 5.x names differ from their SunOS 4.x names. Where possible, these symbolic links point to the device’s SunOS 5.x name rather than to the actual /devices entry.

ucblinks does not remove unneeded compatibility links; these must be removed by hand.

ucblinks should be called each time the system is reconfiguration-booted, after any new SunOS 5.x links that are needed have been created, since the reconfiguration may have resulted in more compatibility names being needed.

In releases prior to SunOS 5.4, ucblinks used a nawk rule-base to construct the SunOS 4.x compatible names. ucblinks no longer uses nawk for the default operation, although nawk rule-bases can still be specified with the -e option. The nawk rule-base equivalent to the SunOS 5.4 default operation can be found in /usr/ucblib/ucblinks.awk.

OPTIONS  
- e rulebase  Specify rulebase as the file containing nawk(1) pattern-action statements.
- r rootdir  Specify rootdir as the directory under which dev and devices will be found, rather than the standard root directory /.

FILES  /usr/ucblib/ucblinks.awk  sample rule-base for compatibility links

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

SEE ALSO  devlinks(1M), disks(1M), ports(1M), tapes(1M), attributes(5)
NAME | ul - do underlining
SYNOPSIS | ul [-i] [-t terminal] [filename...]
DESCRIPTION | ul reads the named filenames (or the standard input if none are given) and translates occurrences of underscores to the sequence which indicates underlining for the terminal in use, as specified by the environment variable TERM. ul uses the /usr/share/lib/terminfo entry to determine the appropriate sequences for underlining. If the terminal is incapable of underlining, but is capable of a standout mode then that is used instead. If the terminal can overstrike, or handles underlining automatically, ul degenerates to cat(1). If the terminal cannot underline, underlining is ignored.
OPTIONS | -t terminal Override the terminal kind specified in the environment. If the terminal cannot underline, underlining is ignored. If the terminal name is not found, no underlining is attempted.
        -i Indicate underlining by a separate line containing appropriate dashes ‘−’; this is useful when you want to look at the underlining which is present in an nroff(1) output stream on a CRT-terminal.
RETURN VALUES | ul returns exit code 1 if the file specified is not found.
FILES | /usr/share/lib/terminfo/*
ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWdoc</td>
</tr>
</tbody>
</table>

SEE ALSO | cat(1), man(1), nroff(1), attributes(5)
BUGS | nroff usually generates a series of backspaces and underlines intermixed with the text to indicate underlining. ul makes attempt to optimize the backward motion.
umask(1)

NAME
umask – get or set the file mode creation mask

SYNOPSIS
/usr/bin/umask [-S] [mask]

sh
umask [000]

\n
csh
umask [000]

ksh
umask [-S] [mask]

DESCRIPTION
The umask utility sets the file mode creation mask of the current shell execution
environment to the value specified by the mask operand. This mask affects the initial
value of the file permission bits of subsequently created files. If umask is called in a
subshell or separate utility execution environment, such as one of the following:
(umask 002)
nohup umask ...
find . -exec umask ...

it does not affect the file mode creation mask of the caller’s environment. For this
reason, the /usr/bin/umask utility cannot be used to change the umask in an
ongoing session. Its usefulness is limited to checking the caller’s umask. To change the
umask of an ongoing session you must use one of the shell builtins.

If the mask operand is not specified, the umask utility writes the value of the invoking
process’s file mode creation mask to standard output.

sh
The user file-creation mode mask is set to 000. The three octal digits refer
to read/write/execute permissions for owner, group, and other, respectively (see
chmod(1), chmod(2), and umask(2)). The value of each specified digit is subtracted
from the corresponding “digit” specified by the system for the creation of a file (see
creat(2)). For example, umask 022 removes write permission for group and other
(files normally created with mode 777 become mode 755; files created with mode 666
become mode 644).

- If 000 is omitted, the current value of the mask is printed.
- umask is recognized and executed by the shell.
- umask can be included in the user’s .profile (see profile(4)) and invoked at
  login to automatically set the user’s permissions on files or directories created.

csh
See the description above for the Bourne shell (sh)umask built-in.

ksh
The user file-creation mask is set to mask. mask can either be an octal number or a
symbolic value as described in chmod(1). If a symbolic value is given, the new umask
value is the complement of the result of applying mask to the complement of
the previous umask value. If mask is omitted, the current value of the mask is printed.

OPTIONS
The following option is supported:
-S Produces symbolic output.
The default output style is unspecified, but will be recognized on a subsequent invocation of `umask` on the same system as a `mask` operand to restore the previous file mode creation mask.

**OPERANDS**

The following operand is supported:

- **mask**: A string specifying the new file mode creation mask. The string is treated in the same way as the `mode` operand described in the `chmod(1)` manual page.

  For a `symbolic_mode` value, the new value of the file mode creation mask is the logical complement of the file permission bits portion of the file mode specified by the `symbolic_mode` string.

  In a `symbolic_mode` value, the permissions `op` characters `+` and `−` are interpreted relative to the current file mode creation mask. `+` causes the bits for the indicated permissions to be cleared in the mask. `−` causes the bits of the indicated permissions to be set in the mask.

  The interpretation of `mode` values that specify file mode bits other than the file permission bits is unspecified.

  The file mode creation mask is set to the resulting numeric value.

  The default output of a prior invocation of `umask` on the same system with no operand will also be recognized as a `mask` operand. The use of an operand obtained in this way is not obsolescent, even if it is an octal number.

**OUTPUT**

When the `mask` operand is not specified, the `umask` utility will write a message to standard output that can later be used as a `umask` `mask` operand.

If `-S` is specified, the message will be in the following format:

```
"u=%s,g=%s,o=%s\n", owner permissions, group permissions, \nother permissions
```

where the three values will be combinations of letters from the set `{r, w, x}`. The presence of a letter will indicate that the corresponding bit is clear in the file mode creation mask.

If a `mask` operand is specified, there will be no output written to standard output.

**EXAMPLES**

**EXAMPLE 1 Using the umask command**

Either of the commands:

```
umask a=rx,ug+w
umask 002
```

sets the mode mask so that subsequently created files have their `S_IWOTH` bit cleared.
EXAMPLE 1 Using the umask command  (Continued)

After setting the mode mask with either of the above commands, the `umask` command can be used to write the current value of the mode mask:

```bash
$ umask
0002
```

The output format is unspecified, but historical implementations use the obsolescent octal integer mode format.

```bash
$ umask -S
u=rwx,g=rwx,o=rx
```

Either of these outputs can be used as the mask operand to a subsequent invocation of the `umask` utility.

Assuming the mode mask is set as above, the command:

```bash
umask g-w
```

sets the mode mask so that subsequently created files have their `S_IWGRP` and `S_IWOTH` bits cleared.

The command:

```bash
umask --w
```

sets the mode mask so that subsequently created files have all their write bits cleared. Notice that `mask` operands `r`, `w`, `x`, or anything beginning with a hyphen (`-`), must be preceded by `--` to keep it from being interpreted as an option.

**ENVIRONMENT VARIABLES**

See `environ(5)` for descriptions of the following environment variables that affect the execution of `umask`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

**EXIT STATUS**

The following exit values are returned:

- `0`  The file mode creation mask was successfully changed, or no `mask` operand was supplied.
- `>0`  An error occurred.

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

**SEE ALSO**

`chmod(1)`, `csh(1)`, `ksh(1)`, `sh(1)`, `chmod(2)`, `creat(2)`, `umask(2)`, `profile(4)`, `attributes(5)`, `environ(5)`
NAME
uname – print name of current system

SYNOPSIS
uname [-aimnprsvX]
uname [-S system_name]

DESCRIPTION
The `uname` utility prints information about the current system on the standard output. When options are specified, symbols representing one or more system characteristics will be written to the standard output. If no options are specified, `uname` prints the current operating system’s name. The options print selected information returned by `uname(2)`, `sysinfo(2)`, or both.

OPTIONS
The following options are supported:

- `-a`
  Prints basic information currently available from the system.

- `-i`
  Prints the name of the hardware implementation (platform).

- `-m`
  Prints the machine hardware name (class). Use of this option is discouraged; use `uname -p` instead. See NOTES section below.

- `-n`
  Prints the nodename (the nodename is the name by which the system is known to a communications network).

- `-p`
  Prints the current host’s ISA or processor type.

- `-r`
  Prints the operating system release level.

- `-s`
  Prints the name of the operating system. This is the default.

- `-S system_name`
  The nodename may be changed by specifying a system name argument. The system name argument is restricted to SYS_NMLN characters. SYS_NMLN is an implementation specific value defined in `<sys/utsname.h>`. Only the super-user is allowed this capability. This change does not persist across reboots of the system. Use `sys-unconfig(1M)` to change a host’s name permanently.

- `-v`
  Prints the operating system version.

- `-X`
  Prints expanded system information, one information element per line, as expected by SCO UNIX. The displayed information includes:

  - system name, node, release, version, machine, and number of CPUs.
  - BusType, Serial, and Users (set to “unknown” in Solaris)
  - OEM# and Origin# (set to 0 and 1, respectively)

EXAMPLES
EXAMPLE 1 Printing the OS name and release level

The following command:

```
example% uname -sr
```
prints the operating system name and release level, separated by one SPACE character.
uname(1)

ENVIRONMENT VARIABLES

SYSV3 This variable is used to override the default behavior of uname. This is necessary to make it possible for some INTERACTIVE UNIX Systems and SCO UNIX programs and scripts to work properly. Many scripts use uname to determine the SYSV3 type or the version of the OS to ensure software is compatible with that OS. Setting SYSV3 to an empty string will make uname print the following default values:

nodename nodename 3.2 2 1386

The individual elements that uname displays can also be modified by setting SYSV3 in the following format:

os,sysname,node,rel,ver,mach

| os          | Operating system (IUS or SCO). |
| sysname     | System name.                   |
| node        | Nodename as displayed by the -n option. |
| rel         | Release level as displayed by the -r option. |
| ver         | Version number as displayed by the -v option. |
| mach        | Machine name as displayed by -m option. |

Do not put spaces between the elements. If an element is omitted, the current system value will be used.

See environ(5) for descriptions of the following environment variables that affect the execution of uname: LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO

arch(1), isalist(1), sys-unconfig(1M), sysinfo(2), uname(2), nodename(4), attributes(5), environ(5)
Independent software vendors (ISVs) and others who need to determine detailed characteristics of the platform on which their software is either being installed or executed should use the `uname` command.

To determine the operating system name and release level, use `uname -sr`. To determine only the operating system release level, use `uname -r`. Notice that operating system release levels are not guaranteed to be in `x.y` format (such as 5.3, 5.4, 5.5, and so forth); future releases could be in the `x.y.z` format (such as 5.3.1, 5.3.2, 5.4.1, and so forth).

In SunOS 4.x releases, the `arch` command was often used to obtain information similar to that obtained by using the `uname` command. The `arch` command output "sun4" was often incorrectly interpreted to signify a SunOS SPARC system. If hardware platform information is desired, use `uname -sp`.

The `arch` and `uname` commands return equivalent values; however, the use of either of these commands by third party programs is discouraged, as is the use of the `arch` command in general. To determine the machine’s Instruction Set Architecture (ISA or processor type), use `uname` with the `-p` option.
unifdef – resolve and remove ifdef-ed lines from C program source

unifdef [-clt] [-D name] [-U name] [-iD name] [-iU name] ... [filename]

unifdef removes ifdef-ed lines from a file while otherwise leaving the file alone. It is smart enough to deal with the nested ifdefs, comments, single and double quotes of C syntax, but it does not do any including or interpretation of macros. Neither does it strip out comments, though it recognizes and ignores them. You specify which symbols you want defined with -D options, and which you want undefined with -U options. Lines within those ifdefs will be copied to the output, or removed, as appropriate. Any ifdef, ifndef, else, and endif lines associated with filename will also be removed.

ifdefs involving symbols you do not specify are untouched and copied out along with their associated ifdef, else, and endif lines.

If an ifdefX occurs nested inside another ifdefX, then the inside ifdef is treated as if it were an unrecognized symbol. If the same symbol appears in more than one argument, only the first occurrence is significant.

unifdef copies its output to the standard output and will take its input from the standard input if no filename argument is given.

The following options are supported:

- c Complement the normal operation. Lines that would have been removed or blanked are retained, and vice versa.
- l Replace “lines removed” lines with blank lines.
- t Plain text option. unifdef refrains from attempting to recognize comments and single and double quotes.
- D name Lines associated with the defined symbol name.
- U name Lines associated with the undefined symbol name.
- iD name Ignore, but print out, lines associated with the defined symbol name. If you use ifdefs to delimit non-C lines, such as comments or code which is under construction, then you must tell unifdef which symbols are used for that purpose so that it will not try to parse for quotes and comments within them.
- iU name Ignore, but print out, lines associated with the undefined symbol name.

The following exit values are returned:

0 Successful operation.
1 Operation failed.

See attributes(5) for descriptions of the following attributes:
See Also:
diff(1), attributes(5)

Diagnostics:
Premature EOF  Inappropriate else or endif.
uniq(1)

NAME uniq – report or filter out repeated lines in a file

SYNOPSIS
uniq [-c | -d | -u] [-f fields] [-s char] [input_file [output_file]]
uniq [-c | -d | -u] [-n] [+ m] [input_file [output_file]]

DESCRIPTION The uniq utility will read an input file comparing adjacent lines, and write one copy of each input line on the output. The second and succeeding copies of repeated adjacent input lines will not be written.

Repeated lines in the input will not be detected if they are not adjacent.

OPTIONS The following options are supported:
- c Precede each output line with a count of the number of times the line occurred in the input.
- d Suppress the writing of lines that are not repeated in the input.
- f fields Ignore the first fields fields on each input line when doing comparisons, where fields is a positive decimal integer. A field is the maximal string matched by the basic regular expression:

```
[[:blank:]]*[[:blank:]]
```

If fields specifies more fields than appear on an input line, a null string will be used for comparison.

- s char Ignore the first chars characters when doing comparisons, where chars is a positive decimal integer. If specified in conjunction with the - f option, the first chars characters after the first fields fields will be ignored. If chars specifies more characters than remain on an input line, a null string will be used for comparison.

- u Suppress the writing of lines that are repeated in the input.

- n Equivalent to - f fields with fields set to n.

+ m Equivalent to - s chars with chars set to m.

OPERANDS The following operands are supported:
input_file A path name of the input file. If input_file is not specified, or if the input_file is -, the standard input will be used.

output_file A path name of the output file. If output_file is not specified, the standard output will be used. The results are unspecified if the file named by output_file is the file named by input_file.

EXAMPLES EXAMPLE 1 Using The uniq Command

The following example lists the contents of the uniq.test file and outputs a copy of the repeated lines.

example$ cat uniq.test
This is a test.

1604 man pages section 1: User Commands • Last Revised 20 Dec 1996
EXAMPLE 1 Using The uniq Command (Continued)

This is a test.
TEST.
Computer.
TEST.
TEST.
Software.

eexample$ uniq -d uniq.test
This is a test.
TEST.
eexample$

The next example outputs just those lines that are not repeated in the uniq.test file.

eexample$ uniq -u uniq.test
TEST.
Computer.
Software.
eexample$

The last example outputs a report with each line preceded by a count of the number of times each line occurred in the file:

eexample$ uniq -c uniq.test
  2 This is a test.
    1 TEST.
    1 Computer.
    2 TEST.
    1 Software.
eexample$

ENVIRONMENT VARIABLES
See environ(5) for descriptions of the following environment variables that affect the execution of uniq: LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS
The following exit values are returned:

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>&gt;0</td>
<td>An error occurred.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

SEE ALSO
comm(1), pack(1), pcat(1), sort(1), uncompress(1), attributes(5), environ(5)
units(1)

NAME units – converts quantities expressed in standard scales to other scales

SYNOPSIS units

DESCRIPTION units converts quantities expressed in various standard scales to their equivalents in other scales. It works interactively in this fashion:

You have:~~inch
You want:~~cm
   * 2.540000e+00
   / 3.937008e-01

A quantity is specified as a multiplicative combination of units optionally preceded by a numeric multiplier. Powers are indicated by suffixed positive integers, division by the usual sign:

You have:~~15 lbs force/in2
You want:~~atm
   * 1.020689e+00
   / 9.797299e-01

units only does multiplicative scale changes; thus it can convert Kelvin to Rankine, but not Celsius to Fahrenheit. Most familiar units, abbreviations, and metric prefixes are recognized, together with a generous leavening of exotica and a few constants of nature including:

pi ratio of circumference to diameter,
c speed of light,
e charge on an electron,
g acceleration of gravity,
force same as g,
mole Avogadro’s number,
water pressure head per unit height of water,
astronomical unit.

Pound is not recognized as a unit of mass; lb is. Compound names are run together, (for example, lightyear). British units that differ from their U.S. counterparts are prefixed thus: brgallon. For a complete list of units, type:

cat /usr/share/lib/unittab

FILES /usr/share/lib/unittab

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
</tbody>
</table>

SEE ALSO: attributes(5)
unix2dos(1)

NAME    unix2dos – convert text file from ISO format to DOS format


DESCRIPTION The unix2dos utility converts ISO standard characters to the corresponding characters in the DOS extended character set.

This command may be invoked from either DOS or SunOS. However, the filenames must conform to the conventions of the environment in which the command is invoked.

If the original file and the converted file are the same, unix2dos will rewrite the original file after converting it.

OPTIONS The following options are supported:

-ascii       Adds carriage returns and converts end of file characters in SunOS format text files to conform to DOS requirements.
-iso         This is the default. Converts ISO standard characters to the corresponding character in the DOS extended character set.
-7           Converts 8 bit SunOS characters to 7 bit DOS characters.

On non-i386 systems, unix2dos will attempt to obtain the keyboard type to determine which code page to use. Otherwise, the default is US. The user may override the code page with one of the following options:

-437         Use US code page
-850         Use multilingual code page
-860         Use Portuguese code page
-863         Use French Canadian code page
-865         Use Danish code page

OPERANDS The following operands are required:

originalfile The original file in ISO format that is being converted to DOS format.
convertedfile The new file in DOS format that has been converted from the original ISO file format.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
</tbody>
</table>

1608  man pages section 1: User Commands  •  Last Revised 14 Sep 2000
The input file you specified does not exist, or you do not have read permission. Check with the SunOS command, `ls -l` (see `ls(1)`).

The output file you specified is either invalid, or you do not have write permission for that file or the directory that contains it. Check also that the drive or diskette is not write-protected.

An error occurred while converting your file, possibly because there is not enough space on the current drive. Check the amount of space on the current drive using the DIR command. Also be certain that the default diskette or drive is write-enabled (not write-protected). Notice that when this error occurs, the original file remains intact.

The program could not perform the final step in converting your file. Your converted file is stored under the name indicated on the second line of this message.
NAME uptime – show how long the system has been up

SYNOPSIS uptime

DESCRIPTION The uptime command prints the current time, the length of time the system has been up, and the average number of jobs in the run queue over the last 1, 5 and 15 minutes. It is, essentially, the first line of a w(1) command.

EXAMPLES Below is an example of the output uptime provides:

```
example% uptime
10:47am up 27 day(s), 50 mins, 1 user, load average: 0.18, 0.26, 0.20
```

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

```
<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>
```

SEE ALSO w(1), who(1), whodo(1M), attributes(5)

NOTES who -b gives the time the system was last booted.
The `users` utility lists the login names of the users currently on the system in a compact, one-line format.

Specifying `filename` tells `users` where to find its information; by default it checks `/var/adm/utmpx`.

Typing `users` is equivalent to typing `who -q`.

**EXAMPLES**

**EXAMPLE 1** Listing current users
```
ex@ple% users
paul  george  ringo
```

**FILES**
```
/var/adm/utmpx
```

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

**SEE ALSO** `who(1), attributes(5)`
NAME  uucp, uulog, uuname – UNIX-to-UNIX system copy

            [-xdebug_level] source-file destination-file

            uulog [-sys] [-f system] [-x] [-number] system

            uuname [-c | -l]

uucp  uucp copies files named by the source-file arguments to the destination-file argument.

uulog  uulog queries a log file of uucp or uuxqt transactions in file
       /var/uucp/.Log/uucico/system or /var/uucp/.Log/uuxqt/system.

uuname  uuname lists the names of systems known to uucp.

uucp  The following options are supported by uucp:

   -c  Do not copy local file to the spool directory for transfer
c      to the remote machine (default).

   -C  Force the copy of local files to the spool directory for
t      transfer.

   -d  Make all necessary directories for the file copy
t      (default).

   -f  Do not make intermediate directories for the file copy.

   -g grade  grade can be either a single letter, number, or a string
            of alphanumeric characters defining a service grade. The
            uuglist command can determine whether it is
            appropriate to use the single letter, number, or a string
            of alphanumeric characters as a service grade. The
            output from the uuglist command will be a list of
            service grades that are available, or a message that says
            to use a single letter or number as a grade of service.

   -j  Print the uucp job identification string on standard
      output. This job identification can be used by uustat
t      to obtain the status of a uucp job or to terminate a
      uucp job. The uucp job is valid as long as the job
      remains queued on the local system.

   -m  Send mail to the requester when the copy is complete.

   -n user  Notify user on the remote system that a file was sent.

   -r  Do not start the file transfer, just queue the job.

   -s file  Report status of the transfer to file. This option is
            accepted for compatibility, but it is ignored because it is
            insecure.
### uucp(1C)

- **debug_level**
  
  Produce debugging output on standard output.
  
  *debug_level* is a number between 0 and 9; as it increases to 9, more detailed debugging information is given.
  
  This option may not be available on all systems.

#### uulog

The following options cause *uulog* to print logging information:

- **-s sys**
  
  Print information about file transfer work involving system *sys*.

- **-f system**
  
  Do a "*tail -f*" of the file transfer log for *system*. (You must hit BREAK to exit this function.)

Other options used in conjunction with the above options are:

- **-x**
  
  Look in the *uuxqt* log file for the given system.

- **-number**
  
  Execute a *tail* command of *number* lines.

#### uuname

The following options are supported by *uuname*:

- **-c**
  
  Display the names of systems known to *cu*. The two lists are the same, unless your machine is using different *Systems* files for *cu* and *uucp*. See the *Sysfiles* file.

- **-l**
  
  Display the local system name.

#### OPERANDS

The source file name may be a path name on your machine, or may have the form:

```
system-name!pathname
```

where *system-name* is taken from a list of system names that *uucp* knows about. *source_file* is restricted to no more than one *system-name*. The destination *system-name* may also include a list of system names such as

```
system-name!system-name!...!system-name!pathname
```

In this case, an attempt is made to send the file, using the specified route, to the destination. Care should be taken to ensure that intermediate nodes in the route are willing to forward information (see *NOTES* below for restrictions).

For C-Shell users, the “!!” character must be surrounded by single quotes (""), or preceded by a backslash (\).

The shell metacharacters ?, *, and [ . . . ] appearing in *pathname* will be expanded on the appropriate system.

Pathnames may be one of the following:

1. An absolute pathname.
A pathname preceded by `~user` where `user` is a login name on the specified
system and is replaced by that user’s login directory.

A pathname preceded by `~/destination` where `destination` is appended to
`/var/spool/uucppublic`. (Note: This destination will be treated as a
filename unless more than one file is being transferred by this request or
the destination is already a directory. To ensure that the destination is a
directory, follow it with a ‘/’. For example `~/dan/` as the destination will
make the directory `/var/spool/uucppublic/dan` if it does not exist
and put the requested file(s) in that directory).

Anything else is prefixed by the current directory.

If the result is an erroneous path name for the remote system, the copy will fail. If the
destination-file is a directory, the last part of the source-file name is used.

Invoking uucp with shell wildcard characters as the remote source-file invokes the
uux(1C) command to execute the uucp command on the remote machine. The remote
uucp command spools the files on the remote machine. After the first session
terminates, if the remote machine is configured to transfer the spooled files to the local
machine, the remote machine will initiate a call and send the files; otherwise, the user
must "call" the remote machine to transfer the files from the spool directory to the local
machine. This call can be done manually using Uutry(1M), or as a side effect of
another uux(1C) or uucp call.

Note that the local machine must have permission to execute the uucp command on
the remote machine in order for the remote machine to send the spooled files.

uucp removes execute permissions across the transmission and gives 0666 read and
write permissions (see chmod(2)).

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the
execution of uucp: LC_COLLATE, LC_CTYPE, LC_MESSAGES, LC_TIME, TZ, and
NLSPATH.

EXIT STATUS

The following exit values are returned:

0       Successful completion.
>0      An error occurred.

FILES

/etc/uucp/*  other data files
/var/spool/uucp spool directories
/usr/lib/uucp/* other program files
/var/spool/uucppublic/* public directory for receiving and sending

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:
For security reasons, the domain of remotely accessible files may be severely restricted. You will probably not be able to access files by path name; ask a responsible person on the remote system to send them to you. For the same reasons you will probably not be able to send files to arbitrary path names. As distributed, the remotely accessible files are those whose names begin /var/spool/uucppublic (equivalent to ~/.)

All files received by uucp will be owned by uucp.

The -m option will only work when sending files or receiving a single file. Receiving multiple files specified by special shell characters ?, &, and [ . . . ] will not activate the -m option.

The forwarding of files through other systems may not be compatible with the previous version of uucp. If forwarding is used, all systems in the route must have compatible versions of uucp.

Protected files and files that are in protected directories that are owned by the requester can be sent by uucp. However, if the requester is root, and the directory is not searchable by "other" or the file is not readable by "other", the request will fail.

Strings that are passed to remote systems may not be evaluated in the same locale as the one in use by the process that invoked uucp on the local system.

Configuration files must be treated as C (or POSIX) locale text files.
**uuencode(1C)**

### NAME
uuencode, uudecode – encode a binary file, or decode its encoded representation

### SYNOPSIS
```
uuencode [source-file] decode_pathname
uudecode [-p] [encoded-file]
```

**uuencode**
uuencode converts a binary file into an encoded representation that can be sent using mail(1). It encodes the contents of source-file, or the standard input if no source-file argument is given. The decode_pathname argument is required. The decode_pathname is included in the encoded file's header as the name of the file into which uudecode is to place the binary (decoded) data. uuencode also includes the permission modes of source-file, (except setuid, setgid, and sticky-bits), so that decode_pathname is recreated with those same permission modes.

**uudecode**
uudecode reads an encoded-file, strips off any leading and trailing lines added by mailer programs, and recreates the original binary data with the filename and the mode specified in the header.

The encoded file is an ordinary portable character set text file; it can be edited by any text editor. It is best only to change the mode or decode_pathname in the header to avoid corrupting the decoded binary.

**uudecode**
```
-p          decode encoded-file and send it to standard output. This allows uudecode to be used in a pipeline.
```

### uuencode
The following operands are supported by uuencode:
```
decode_pathname The pathname of the file into which the uudecode utility will place the decoded file. If there are characters in decode_pathname that are not in the portable filename character set, the results are unspecified.
source-file A pathname of the file to be encoded.
```

### uudecode
The following operand is supported by uudecode:
```
encoded-file The pathname of a file containing the output of uuencode.
```

### USAGE
See largefile(5) for the description of the behavior of uuencode and uudecode when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

### ENVIRONMENT VARIABLES
See environ(5) for descriptions of the following environment variables that affect the execution of uuencode and uudecode: LC_CTYPE, LC_MESSAGES, and NLSPATH.

### stdout
The standard output is a text file (encoded in the character set of the current locale) that begins with the line:
```
"begin%s%s\n", < mode >, decode_pathname
```

and ends with the line:
```
end\n```
In both cases, the lines have no preceding or trailing blank characters.

The algorithm that is used for lines in between begin and end takes three octets as input and writes four characters of output by splitting the input at six-bit intervals into four octets, containing data in the lower six bits only. These octets are converted to characters by adding a value of 0x20 to each octet, so that each octet is in the range 0x20–0x5f, and then it is assumed to represent a printable character. It then will be translated into the corresponding character codes for the codeset in use in the current locale. (For example, the octet 0x41, representing A, would be translated to A in the current codeset, such as 0xc1 if it were EBCDIC.)

Where the bits of two octets are combined, the least significant bits of the first octet are shifted left and combined with the most significant bits of the second octet shifted right. Thus the three octets A, B, C are converted into the four octets:

\[
\begin{align*}
0x20 + (( A >> 2 ) & 0x3F) \\
0x20 + (((A << 4) ((B >> 4) & 0xF)) & 0x3F) \\
0x20 + (((B << 2) ((C >> 6) & 0x3)) & 0x3F) \\
0x20 + (( C ) & 0x3F)
\end{align*}
\]

These octets are then translated into the local character set.

Each encoded line contains a length character, equal to the number of characters to be decoded plus 0x20 translated to the local character set as described above, followed by the encoded characters. The maximum number of octets to be encoded on each line is 45.

EXIT STATUS
The following exit values are returned:

0 Successful completion.

>0 An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
</tbody>
</table>

SEE ALSO
mail(1), mailx(1), uucp(1C), uux(1C), attributes(5), largefile(5)

NOTES
The encoded file’s size is expanded by 35% (3 bytes become 4, plus control information), causing it to take longer to transmit than the equivalent binary.

The user on the remote system who is invoking uudecode (typically uucp) must have write permission on the file specified in the decode pathname.

If you uuencode then uudecode a file in the same directory, you will overwrite the original file.
NAME  uuglist – print the list of service grades that are available on this UNIX system
SYNOPSIS  uuglist [-u]
DESCRIPTION  uuglist prints the list of service grades that are available on the system to use with the -g option of uucp(1C) and uux(1C).
OPTIONS  -u  List the names of the service grades that the user is allowed to use with the -g option of the uucp and uux commands.
FILES  /etc/uucp/Grades contains the list of service grades
ATTRIBUTES  See attributes(5) for descriptions of the following attributes:

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWbnuu</td>
</tr>
</tbody>
</table>

SEE ALSO  uucp(1C), uux(1C), attributes(5)
NAME

uustat - uucp status inquiry and job control

SYNOPSIS

uustat [ [-m] | [-p] | [-q] | [ -k jobid [-n]] | [ -r jobid [-n]] ]

uustat [-a] [-s system [-j]] [-u user] [-S qtric]

uustat -t system [-c] [-d number]

DESCRIPTION

uustat functions in the following three areas:

1.) Displays the general status of, or cancels, previously specified uucp commands.

2.) Provides remote system performance information, in terms of average transfer rates or average queue times.

3.) Provides general remote system-specific and user-specific status of uucp connections to other systems.

General Status

These options obtain general status of, or cancel, previously specified uucp commands:

- **-a** List all jobs in queue.
- **-j** List the total number of jobs displayed. The -j option can be used in conjunction with the -a or the -s option.
- **-k jobid** Kill the uucp request whose job identification is jobid. The killed uucp request must belong to the user issuing the uustat command unless the user is the super-user or uucp administrator. If the job is killed by the super-user or uucp administrator, electronic mail is sent to the user.
- **-m** Report the status of accessibility of all machines.
- **-n** Suppress all standard output, but not standard error. The -n option is used in conjunction with the -k and -r options.
- **-p** Execute the command `ps -flp` for all the process-ids that are in the lock files.
- **-q** List the jobs queued for each machine. If a status file exists for the machine, its date, time and status information are reported. In addition, if a number appears in parentheses next to the number of C or X files, it is the age in days of the oldest C./X. file for that system. The Retry field represents the number of hours until the next possible call. The Count is the number of failure attempts. Note: For systems with a moderate number of outstanding jobs, this could take 30 seconds or more of real-time to execute. An example of the output produced by the -q option is:

```
eagle 3C 04/07-11:07 NO DEVICES AVAILABLE
mh3bs3 2C 07/07-10:42 SUCCESSFUL
```

User Commands 1619
This indicates the number of command files that are waiting for each system. Each command file may have zero or more files to be sent (zero means to call the system and see if work is to be done). The date and time refer to the previous interaction with the system followed by the status of the interaction.

- \texttt{-r jobid} Rejuvenate \texttt{jobid}. The files associated with \texttt{jobid} are touched so that their modification time is set to the current time. This prevents the cleanup daemon from deleting the job until the jobs' modification time reaches the limit imposed by the daemon.

These options provide remote system performance information, in terms of average transfer rates or average queue times; the \texttt{-c} and \texttt{-d} options can only be used in conjunction with the \texttt{-t} option:

- \texttt{-t system} Report the average transfer rate or average queue time for the past 60 minutes for the remote \texttt{system}. The following parameters can only be used with this option:

- \texttt{-c} Average queue time is calculated when the \texttt{-c} parameter is specified and average transfer rate when \texttt{-c} is not specified. For example, the command:

  \[ \texttt{example\% uustat -teagle -d50 -c} \]

  produces output in the following format:

  \[ \texttt{average queue time to eagle for last 50 minutes: 5 seconds} \]

  The same command without the \texttt{-c} parameter produces output in the following format:

  \[ \texttt{average transfer rate with eagle for last 50 minutes: 2000.88 bytes/sec} \]

- \texttt{-d number} \texttt{number} is specified in minutes. Used to override the 60 minute default used for calculations. These calculations are based on information contained in the optional performance log and therefore may not be available. Calculations can only be made from the time that the performance log was last cleaned up.

These options provide general remote system-specific and user-specific status of \texttt{uucp} connections to other systems. Either or both of the following options can be specified with \texttt{uustat}. The \texttt{-j} option can be used in conjunction with the \texttt{-s} option to list the total number of jobs displayed:

- \texttt{-s system} Report the status of all \texttt{uucp} requests for remote system \texttt{system}. 

\[ \text{uustat(1C)} \]
-user Report the status of all uucp requests issued by user.

Output for both the -s and -u options has the following format:

```
eagleN1bd7 4/07-11:07  S  eagle  dan  522  /home/dan/A
eagleC1bd8 4/07-11:07  S  eagle  dan  59   D.3b2a12ce4924
 4/07-11:07  S  eagle  dan  rmail  mike
```

With the above two options, the first field is the jobid of the job. This is followed by the date/time. The next field is an S if the job is sending a file or an R if the job is requesting a file. The next field is the machine where the file is to be transferred. This is followed by the user-id of the user who queued the job. The next field contains the size of the file, or in the case of a remote execution (rmail is the command used for remote mail), the name of the command. When the size appears in this field, the file name is also given. This can either be the name given by the user or an internal name (for example, D.3b2a12ce4924) that is created for data files associated with remote executions (rmail in this example).

-Sqr Report the job state:

```
q for queued jobs
r for running jobs
i for interrupted jobs
c for completed jobs
```

A job is queued if the transfer has not started. A job is running when the transfer has begun. A job is interrupted if the transfer began but was terminated before the file was completely transferred. A completed job is a job that successfully transferred. The completed state information is maintained in the accounting log, which is optional and therefore may be unavailable. The parameters can be used in any combination, but at least one parameter must be specified. The -S option can also be used with -s and -u options. The output for this option is exactly like the output for -s and -u except that the job states are appended as the last output word.

Output for a completed job has the following format:

```
eagleC1bd3 completed
```

When no options are given, uustat writes to standard output the status of all uucp requests issued by the current user.

**ENVIRONMENT VARIABLES**
See environ(5) for descriptions of the following environment variables that affect the execution of uustat: LC_CTYPE, LC_MESSAGES, LC_TIME, TZ, and NLSPATH.

**EXIT STATUS**
The following exit values are returned:

```
0    Successful completion.
>0   An error occurred.
```

**FILES**
/var/spool/uucp/*  spool directories
uustat(1C)

/var/uucp/.Admin/account accounting log
/var/uucp/.Admin/perflog performance log

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWbnuu</td>
</tr>
</tbody>
</table>

SEE ALSO

uucp(1C), attributes(5)

DIAGNOSTICS

The -t option produces no message when the data needed for the calculations is not being recorded.

NOTES

After the user has issued the uucp request, if the file to be transferred is moved, deleted or was not copied to the spool directory (-C option) when the uucp request was made, uustat reports a file size of −99999. This job will eventually fail because the file(s) to be transferred can not be found.
### NAME
uuto, uupick – public UNIX-to-UNIX system file copy

### SYNOPSIS
**uuto**
```
-uuto [ -mp ] source-file... destination
```

**uupick**
```
-uupick [ -s system ]
```

**uuto**
 sends *source-file* to *destination*.  **uuto** uses the **uucp(1C)** facility to send files, while it allows the local system to control the file access. A *source-file* name is a path name on your machine.  Destination has the form:

```
system[system] ... !user
```

where *system* is taken from a list of system names that **uucp** knows about.  *User* is the login name of someone on the specified system.

The files (or sub-trees if directories are specified) are sent to **PUBDIR** on **system**, where **PUBDIR** is a public directory defined in the **uucp** source. By default, this directory is `/var/spool/uucppublic`.  Specifically the files are sent to

```
PUBDIR/receive/user/mysystem/files.
```

The recipient is notified by **mail(1)** of the arrival of files.

**uupick**
 uupick accepts or rejects the files transmitted to the user. Specifically, **uupick** searches **PUBDIR** for files destined for the user.  For each entry (file or directory) found, the following message is printed on standard output:

```from system sysname: [file file-name] [dir dirname] ?
```

**uupick** then reads a line from standard input to determine the disposition of the file:

- `<new-line>`: Go to next entry.
- `d`: Delete the entry.
- `m [ dir ]`: Move the entry to named directory *dir*.  If *dir* is not specified as a complete path name (in which `$HOME` is legitimate), a destination relative to the current directory is assumed.  If no destination is given, the default is the current directory.
- `a [ dir ]`: Same as `m` above, except it moves all the files sent from *system*.
- `p`: Print the content of the file.
- `q`: Stop.
- `EOT (control-d)`: Same as `q`.
- `!command`: Escape to the shell to do command.
* Print a command summary.

**uuto**

The following options are supported by *uuto*:

- `-m` Send mail to the sender when the copy is complete.
- `-p` Copy the source file into the spool directory before transmission.

**uupick**

The following option is supported by *uupick*:

- `-s system` Search only the PUBDIR for files sent from *system*.

**OPERANDS**

The following operands are supported for *uuto*:

- `destination` A string of the form:

  ```
  system-name ! user
  ```

  where *system-name* is taken from a list of system names that *uucp* knows about; see *uname*. The argument *user* is the login name of someone on the specified system. The destination *system-name* can also be a list of names such as

  ```
  system-name ! system-name ! ... ! system-name ! user
  ```

  in which case, an attempt is made to send the file via the specified route to the destination. Care should be taken to ensure that intermediate nodes in the route are willing to forward information.

- `source-file` A pathname of a file on the local system to be copied to *destination*.

**ENVIRONMENT VARIABLES**

See *environ*(5) for descriptions of the following environment variables that affect the execution of *uuto* and *uupick*: `LC_TYPE`, `LC_MESSAGES`, and `NLSPATH`.

**EXIT STATUS**

The following exit values are returned:

- `0` Successful completion.
- `>0` An error occurred.

**FILES**

- `PUBDIR` `/var/spool/uucppublic` Public directory

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWbnuu</td>
</tr>
</tbody>
</table>

**SEE ALSO**

- `mail(1)`, `uucp(1C)`, `uustat(1C)`, `uux(1C)`, `uucleanup(1M)`, *attributes*(5)
In order to send files that begin with a dot (for instance, `.profile`), the files must be qualified with a dot. For example, the following files are correct:

```
.profile .prof* .profil?
```

The following files are incorrect:

```
*prof* ?profile
```
**NAME**
uux – UNIX-to-UNIX system command execution

**SYNOPSIS**
uux [-] [-bcCjnprz] [-name] [-ggrade] [-sfilename] [-xdebug_level]
   command-string

**DESCRIPTION**
uux will gather zero or more files from various systems, execute a command on a
specified system and then send standard output to a file on a specified system.

Note: For security reasons, most installations limit the list of commands executable on
behalf of an incoming request from uux, permitting only the receipt of mail (see
mail(1)). (Remote execution permissions are defined in /etc/uucp/Permissions.)

The **command-string** is made up of one or more arguments that look like a shell
command line, except that the command and file names may be prefixed by
system-name!. A null system-name is interpreted as the local system.

File names may be one of the following:
   ■ An absolute path name.
   ■ A path name preceded by ~xxx, where xxx is a login name on the specified system
      and is replaced by that user’s login directory.

Anything else is prefixed by the current directory.

As an example, the command:

eample% uux "!diff sys1!/home/dan/filename1
   sys2!/a4/dan/filename2 > !~/dan/filename.diff"

will get the filename1 and filename2 files from the “sys1” and “sys2” machines, execute
a diff(1) command and put the results in filename.diff in the local PUBDIR/dan/
directory. PUBDIR is a public directory defined in the uucp source. By default, this
directory is /var/spool/uucppublic.

Any special shell characters such as <, >, ; , | should be quoted either by quoting
the entire command-string, or quoting the special characters as individual arguments.
The redirection operators >>, <<, > | and >& cannot be used.

uux will attempt to get all appropriate files to the specified system where they will be
processed. For files that are output files, the file name must be escaped using
parentheses. For example, the command:

eample% uux "a!cut -f1 b!/usr/filename > c!/usr/filename"

gets "/usr/filename" from system "b" and sends it to system "a", performs a cut
command on that file and sends the result of the cut command to system "c".

uux will notify you if the requested command on the remote system was disallowed.
This notification can be turned off by the -n option. The response comes by remote
mail from the remote machine.
## OPTIONS

**-**  
The standard input to `uux` is made the standard input to the `command-string`.

**-aname**  
Use `name` as the user job identification replacing the initiator user-id. (Notification will be returned to user-id `name`.)

**-b**  
Return whatever standard input was provided to the `uux` command if the exit status is non-zero.

**-c**  
Do not copy local file to the spool directory for transfer to the remote machine (default).

**-C**  
Force the copy of local files to the spool directory for transfer.

**-g grade**  
`grade` can be either a single letter, number, or a string of alphanumeric characters defining a service grade. The `uuglist(1C)` command determines whether it is appropriate to use the single letter, number, or a string of alphanumeric characters as a service grade. The output from the `uuglist` command will be a list of service grades that are available or a message that says to use a single letter or number as a grade of service.

**-j**  
Output the jobid string on the standard output which is the job identification. This job identification can be used by `uustat(1C)` to obtain the status or terminate a job.

**-n**  
Do not notify the user if the command fails.

**-p**  
Same as `-`: The standard input to `uux` is made the standard input to the `command-string`.

**-r**  
Do not start the file transfer, just queue the job.

**-s filename**  
Report status of the transfer in `filename`. This option is accepted for compatibility, but it is ignored because it is insecure.

**-x debug_level**  
Produce debugging output on the standard output. `debug_level` is a number between 0 and 9; as it increases to 9, more detailed debugging information is given.

**-z**  
Send success notification to the user.

## ENVIRONMENT VARIABLES

See `environ(5)` for descriptions of the following environment variables that affect the execution of `uux`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

## EXIT STATUS

The following exit values are returned:

<table>
<thead>
<tr>
<th>Exit Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
</tbody>
</table>
An error occurred.

**FILES**
- `/etc/uucp/*` other data and programs
- `/etc/uucp/Permissions` remote execution permissions
- `/usr/lib/uucp/*` other programs
- `/var/spool/uucp` spool directories

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWbnuu</td>
</tr>
</tbody>
</table>

**SEE ALSO**
cut(1), mail(1), uucp(1C), uuglist(1C), uustat(1C), attributes(5)

**NOTES**
The execution of commands on remote systems takes place in an execution directory known to the uucp system.

All files required for the execution will be put into this directory unless they already reside on that machine. Therefore, the simple file name (without path or machine reference) must be unique within the uux request. The following command will NOT work:

```bash
example% uux "a!diff b!/home/dan/xyz c!/home/dan/xyz > !xyz.diff"
```

But the command:

```bash
example% uux "a!diff a!/home/dan/xyz c!/home/dan/xyz > !xyz.diff"
```

will work. (If `diff` is a permitted command.)

Protected files and files that are in protected directories that are owned by the requester can be sent in commands using `uux`. However, if the requester is root, and the directory is not searchable by "other", the request will fail.

The following restrictions apply to the shell pipeline processed by `uux`:

- In gathering files from different systems, pathname expansion in not performed by `uux`. Thus, a request such as
  ```bash
  uux "c89 remsys!-/*.c"
  ```
  would attempt to copy the file named literally `*.c` to the local system.
- Only the first command of a shell pipeline may have a `system-name!`. All other commands are executed on the system of the first command.
- The use of the shell metacharacter `*` will probably not do what you want it to do.
- The shell tokens `<<` and `>>` are not implemented.
The redirection operators >>, <<, >| and >& cannot be used.

The reserved word ! cannot be used at the head of the pipeline to modify the exit status.

Alias substitution is not performed.
The vacation utility automatically replies to incoming mail. The installation consists of an interactive program which sets up vacation’s basic configuration.

To install vacation, type it with no arguments on the command line. The program creates a .vacation.msg file, which contains the message that is automatically sent to all senders when vacation is enabled, and starts an editor for you to modify the message. (See USAGE section.) Which editor is invoked is determined by the VISUAL or EDITOR environment variable, or vi(1) if neither of those environment variables are set.

A .forward file is also created if one does not exist in your home directory. Once created, the .forward file will contain a line of the form:

`\username, ":/usr/bin/vacation username"

One copy of an incoming message is sent to the username and another copy is piped into vacation.

If a .forward file is present in your home directory, it will ask whether you want to remove it, which disables vacation and ends the installation.

The program automatically creates .vacation.pag and .vacation.dir, which contain a list of senders when vacation is enabled.

The presence of the .forward file determines whether or not vacation is disabled or enabled. To disable vacation, remove the .forward file, or move it to a new name.

The -I option clears the vacation log files, .vacation.pag and .vacation.dir, erasing the list of senders from a previous vacation session. (See OPTIONS section.)

vacation provides configuration options that are not part of the installation, these being -a, -f, -j, -m, -s, and -t. (See OPTIONS section.)

The following options are supported:

- **-I** 
  Initializes the .vacation.pag and .vacation.dir files and enables vacation. If the -I flag is not specified, and a user argument is given, vacation reads the first line from the standard input (for a From: line, no colon). If absent, it produces an error message.

Options -a, -f, -j, -m, -t, and -s are configuration options to be used in conjunction with vacation in the .forward file, not on the command line. For example,
vacation(1)

\username, "/usr/bin/vacation -t1m \username" repeats replies to the sender every minute.

-\a \alias Indicates that \alias is one of the valid aliases for the user running vacation, so that mail addressed to that alias generates a reply.

-\f \file Uses \file instead of .vacation as the base name for the database file.

-\j Does not check whether the recipient appears in the To: or the Cc: line. Warning: use of this option can result in vacation replies being sent to mailing lists and other inappropriate places; its use is therefore strongly discouraged.

-\m \file Uses \file instead of .vacation.msg as the message to send for the reply.

-\s \sender Replies to \sender instead of the value read from the UNIX From line of the incoming message.

-\t N Changes the interval between repeat replies to the same sender. The default is 1 week. A trailing s, m, h, d, or w scales N to seconds, minutes, hours, days, or weeks, respectively.

Files .vacation.msg should include a header with at least a Subject: line (it should not include a To: line). For example:

Subject: I am on vacation
I am on vacation until July 22. If you have something urgent, please contact Joe Jones (jones@fB0).
   --John

If the string $\SUBJECT$ appears in the .vacation.msg file, it is replaced with the subject of the original message when the reply is sent; thus, a .vacation.msg file such as

Subject: I am on vacation
I am on vacation until July 22.
Your mail regarding "$\SUBJECT$" will be read when I return.
If you have something urgent, please contact Joe Jones (jones@fB0).
   --John

will include the subject of the message in the reply.

No message is sent if the To: or the Cc: line does not list the user to whom the original message was sent or one of a number of aliases for them, if the initial From line includes the string --REQUEST, or if a Precedence: bulk or Precedence: junk line is included in the header.

vacation will also not respond to mail from either postmaster or Mailer-Daemon.
A list of senders is kept in the dbm format files .vacation.pag and .vacation.dir in your home directory. These files are dbm files and cannot be viewed directly with text editors.

ATTRIBUTES

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO

vi(1), sendmail(1M), dbm(3UCB), getusershell(3C), aliases(4), shells(4), attributes(5)
vc(1)

NAME
vc – version control

SYNOPSIS
vc [-a] [-t] [-c char] [-s] [keyword=value... keyword=value]

DESCRIPTION
This command is obsolete and will be removed in the next release.

The vc command copies lines from the standard input to the standard output under control of its arguments and of “control statements” encountered in the standard input. In the process of performing the copy operation, user-declared keywords may be replaced by their string value when they appear in plain text and/or control statements.

The copying of lines from the standard input to the standard output is conditional, based on tests (in control statements) of keyword values specified in control statements or as vc command arguments.

A control statement is a single line beginning with a control character, except as modified by the -t keyletter (see below). The default control character is colon (:), except as modified by the -c keyletter (see below). Input lines beginning with a backslash (\) followed by a control character are not control lines and are copied to the standard output with the backslash removed. Lines beginning with a backslash followed by a non-control character are copied in their entirety.

A keyword is composed of 9 or less alphanumerics; the first must be alphabetic. A value is any ASCII string that can be created with ed; a numeric value is an unsigned string of digits. Keyword values may not contain blanks or tabs.

Replacement of keywords by values is done whenever a keyword surrounded by control characters is encountered on a version control statement. The -a keyletter (see below) forces replacement of keywords in all lines of text. An uninterpreted control character may be included in a value by preceding it with \. If a literal \ is desired, then it too must be preceded by \\.

OPTIONS
The following options are supported:

-a Forces replacement of keywords surrounded by control characters with their assigned value in all text lines and not just in vc statements.

-t All characters from the beginning of a line up to and including the first tab character are ignored for the purpose of detecting a control statement. If a control statement is found, all characters up to and including the tab are discarded.

-c char Specifies a control character to be used in place of the “:” default.

-s Silences warning messages (not error) that are normally printed on the diagnostic output.

vc recognizes the following version control statements:

:dcl keyword[, ..., keyword] Declare keywords. All keywords must be declared.
:asg keyword=value

Assign values to keywords. An asg statement overrides the assignment for the corresponding keyword on the vc command line and all previous asg statements for that keyword. Keywords that are declared but are not assigned values have null values.

:if condition
...
:end

Skip lines of the standard input. If the condition is true, all lines between the if statement and the matching end statement are copied to the standard output. If the condition is false, all intervening lines are discarded, including control statements.

Note: Intervening if statements and matching end statements are recognized solely for the purpose of maintaining the proper if-end matching.

The syntax of a condition is:

<cond> ::= [ "not" ] <or>
<or> ::= <and> | <and> "|" <or>
<and> ::= <exp> | <exp> "&" <and>
<exp> ::= "(" <or> ")" | <value> <op> <value>
<op> ::= "=" | "!=" | "<" |">
<value> ::= <arbitrary ASCII string> | <numeric string>

The available operators and their meanings are:

= equal
!= not equal
& and
| or
> greater than
< less than
() used for logical groupings
not may only occur immediately after the if, and when present, inverts the value of the entire condition

The > and < operate only on unsigned integer values (for example, :012 > 12 is false). All other operators take strings as arguments (for example, :012 != 12 is true).

The precedence of the operators (from highest to lowest) is:

= != > < all of equal precedence
&
|

Parentheses may be used to alter the order of precedence.

Values must be separated from operators or parentheses by at least one blank or tab.

::text Replace keywords on lines that are copied to the standard output. The two leading control characters are removed, and keywords surrounded by control characters in text are replaced by their value before the line is copied to the output file. This action is independent of the -a keyletter.

:on Turn on or off keyword replacement on all lines.
:off

:ctl char Change the control character to char.
:msg message Print message on the diagnostic output.
:err message Print message followed by:

ERROR: err statement on line ...

(vc)
vc(1)

on the diagnostic output. vc halts execution, and returns an exit code of 1.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsprot</td>
</tr>
</tbody>
</table>

SEE ALSO ed(1), attributes(5)
vgrind(1)

NAME
vgrind – grind nice program listings

SYNOPSIS
vgrind [-2fntwx] [-d defss-file] [-h header] [-1 language] [-s n]
[-o pagelist] [-P printer] [-T output-device] filename...

DESCRIPTION
The vgrind utility formats the program sources named by the filename arguments in a nice style using troff(1). Comments are placed in italics, keywords in bold face, and as each function is encountered its name is listed on the page margin.

vgrind runs in two basic modes, filter mode or regular mode. In filter mode, vgrind acts as a filter in a manner similar to tbl(1). The standard input is passed directly to the standard output except for lines bracketed by the troff-like macros:

.vs starts processing
.ve ends processing

These lines are formatted as described above. The output from this filter can be passed to troff for output. There need be no particular ordering with eqn(1) or tbl(1).

In regular mode, vgrind accepts input filenames, processes them, and passes them to troff for output. Use a hyphen ('-') to specify standard input; otherwise, vgrind will exit without attempting to read from the standard input. Filenames must be specified after all other option arguments.

In regular mode, if the -t or -P option is specified, the output is:

- emitted (in troff format) to stdout if the -t option is specified.
- printed (as PostScript) to the named printer if the -P option is specified.

Otherwise, the output is:

- printed (as PostScript) on the system default printer, if one is defined, and the command’s stdout is a tty.
- emitted (as PostScript) to stdout if it is not a tty (that is, if stdout is a pipe or a redirect to a file).

In both modes, vgrind passes any lines beginning with a decimal point without conversion.

OPTIONS
The following options are supported:

-2 Produces two-column output. Specifying this option changes the default point size to 8 (as if the -s8 option were supplied). It also arranges for output to appear in landscape mode.

-ef Forces filter mode.

-n Does not make keywords boldface.

-w Considers TAB characters to be spaced four columns apart instead of the usual eight.
vgrind(1)

-x Outputs the index file in a “pretty” format. The index file itself is produced whenever vgrind is run with a file called index that is present in the current directory. The index of function definitions can then be run off by giving vgrind the -x option and the file index as argument.

-a defs-file Specifies an alternate language definitions file (default is /usr/lib/vgrindefs).

-h header Specifies a header to appear in the center of every output page. Use quotes to specify headers with embedded spaces.

-l language Specifies the language to use. Among the languages currently known are: Bourne shell (-lsh), C (-lc, the default), C++ (-lc++), C shell (-lcsh), emacs MLisp (-lml), FORTRAN (-lf), Icon (-lI), ISP (-i), LDL (-lLDL), Model (-lm), Pascal (-lp), and RATFOR (-lr).

-P printer Sends output to the named printer.

-s n Specifies a point size to use on output (exactly the same as the argument of a troff.ps point size request).

vgrind passes the following options to the formatter specified by the TROFF environment variable. See ENVIRONMENT VARIABLES.

-t Similar to the same option in troff; that is, formatted text goes to the standard output.

-pagelist Prints only those pages whose page numbers appear in the comma-separated pagelist of numbers and ranges. A range N–M means pages N through M; an initial -N means from the beginning to page N; and a final N– means from N to the end.

-T output-device Formats output for the specified output-device.

**OPERANDS**

The following operand is supported:

filename Name of the program source to be processed by vgrind. Use ‘-’ to specify the standard input.

**ENVIRONMENT VARIABLES**

In regular mode, vgrind feeds its intermediate output to the text formatter given by the value of the TROFF environment variable, or to /usr/bin/troff if this variable is not defined in the environment. This mechanism allows for local variations in troff’s name.

**FILES**

index File where source for index is created

/usr/lib/vgrindefs Language descriptions
ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWdoc</td>
</tr>
</tbody>
</table>

SEE ALSO

csh(1), ctags(1), eqn(1), tbl(1), troff(1), attributes(5), vgrindefs(5)

BUGS

vgrind assumes that a certain programming style is followed:

C
Function names can be preceded on a line only by SPACE, TAB, or an asterisk (*). The parenthesized arguments must also be on the same line.

FORTRAN
Function names need to appear on the same line as the keywords function or subroutine.

MLisp
Function names should not appear on the same line as the preceding defun.

Model
Function names need to appear on the same line as the keywords is beginproc.

Pascal
Function names need to appear on the same line as the keywords function or procedure.

If these conventions are not followed, the indexing and marginal function name comment mechanisms will fail.

More generally, arbitrary formatting styles for programs usually give unsightly results. To prepare a program for vgrind output, use TAB rather than SPACE characters to align source code properly, since vgrind uses variable width fonts.

The mechanism of ctags(1) in recognizing functions should be used here.

The -w option is annoying, but there is no other way to achieve the desired effect.

The macros defined in tmac.vgrind do not coexist gracefully with those of other macro packages, making filter mode difficult to use effectively.

vgrind does not process certain special characters in csh(1) scripts correctly.
The tmac.vgrind formatting macros wire in the page height and width used in two-column mode, effectively making two column output useless for paper sizes other than the standard American size of 8.5 inches by 11 inches. For other paper sizes, it is necessary to edit the size values given in tmac.vgrind. A better solution would be to create a troff output device specification intended specifically for landscape output and record size information there.
vi (visual) is a display-oriented text editor based on an underlying line editor ex. It is possible to use the command mode of ex from within vi and to use the command mode of vi from within ex. The visual commands are described on this manual page; how to set options (like automatically numbering lines and automatically starting a new output line when you type carriage return) and all ex line editor commands are described on the ex(1) manual page.

When using vi, changes you make to the file are reflected in what you see on your terminal screen. The position of the cursor on the screen indicates the position within the file.

The view invocation is the same as vi except that the readonly flag is set.

The vedit invocation is intended for beginners. It is the same as vi except that the report flag is set to 1, the showmode and novice flags are set, and magic is turned off. These defaults make it easier to learn how to use vi.

The following invocation options are interpreted by vi (previously documented options are discussed in the NOTES section of this manual page):

- | -s  Suppress all interactive user feedback. This is useful when processing editor scripts.

-C  Encryption option; same as the -x option, except that vi simulates the C command of ex. The C command is like the X command of ex, except that all text read in is assumed to have been encrypted.

-1  Set up for editing LISP programs.

-L  List the name of all files saved as the result of an editor or system crash.
### vi(1)

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-x filename</code></td>
<td>Edit <code>filename</code> after an editor or system crash. (Recovering the version of <code>filename</code> that was in the buffer when the crash occurred.)</td>
</tr>
<tr>
<td><code>-R</code></td>
<td>Readonly mode; the <code>readonly</code> flag is set, preventing accidental overwriting of the file.</td>
</tr>
<tr>
<td><code>-S</code></td>
<td>This option is used in conjunction with the <code>-t tag</code> option to tell <code>vi</code> that the tags file may not be sorted and that, if the binary search (which relies on a sorted tags file) for <code>tag</code> fails to find it, the much slower linear search should also be done. Since the linear search is slow, users of large tags files should ensure that the tags files are sorted rather than use this flag. Creation of tags files normally produces sorted tags files. See <code>ctags(1)</code> for more information on tags files.</td>
</tr>
<tr>
<td><code>-t tag</code></td>
<td>Edit the file containing the tag, <code>tag</code>, and position the editor at its definition.</td>
</tr>
<tr>
<td><code>-v</code></td>
<td>Start up in display editing state using <code>vi</code>. You can achieve the same effect by simply typing the <code>vi</code> command itself.</td>
</tr>
<tr>
<td><code>-V</code></td>
<td>Verbose. When <code>ex</code> commands are read by means of standard input, the input will be echoed to standard error. This may be useful when processing <code>ex</code> commands within shell scripts.</td>
</tr>
<tr>
<td><code>-wmn</code></td>
<td>Set the default window size to <code>n</code>. This is useful when using the editor over a slow speed line.</td>
</tr>
<tr>
<td><code>-x</code></td>
<td>Encryption option; when used, <code>vi</code> simulates the <code>X</code> command of <code>ex</code> and prompts the user for a key. This key is used to encrypt and decrypt text using the algorithm of the <code>crypt</code> command. The <code>X</code> command makes an educated guess to determine whether text read in is encrypted or not. The temporary buffer file is encrypted also, using a transformed version of the key typed in for the <code>-x</code> option. If an empty encryption key is entered (that is, if the return key is pressed right after the prompt), the file will not be encrypted. This</td>
</tr>
</tbody>
</table>
+command | -c command

Begin editing by executing the specified editor command (usually a search or positioning command).

If both the -t tag and the -c command options are given, the -t tag option will be processed first. That is, the file containing tag is selected by -t and then the command is executed.

The following operands are supported:

filename A file to be edited.

vi Modes

Command Normal and initial mode. Other modes return to command mode upon completion. ESC (escape) is used to cancel a partial command.

Input Entered by setting any of the following options:

A i I o O c C s S R Arbitrary text may then be entered. Input mode is normally terminated with the ESC character, or, abnormally, with an interrupt.

Last line Reading input for : / ? or !. Terminate by typing a carriage return. An interrupt cancels termination.

Sample commands In the descriptions, CR stands for carriage return and ESC stands for the escape key.

<- down-arrow arrow keys move the cursor
up-arrow
h j k l same as arrow keys
i text ESC insert text
cw new ESC change word to new
ea new ESC pluralize word (end of word; append s; escape from input state)
x delete a character
dw delete a word
dd delete a line
3dd delete 3 lines
u undo previous change
## vi(1)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZZ</td>
<td>exit vi, saving changes</td>
</tr>
<tr>
<td>:q!CR</td>
<td>quit, discarding changes</td>
</tr>
<tr>
<td>/textCR</td>
<td>search for text</td>
</tr>
<tr>
<td>^U ^D</td>
<td>scroll up or down</td>
</tr>
<tr>
<td>:cmdCR</td>
<td>any ex or ed command</td>
</tr>
</tbody>
</table>

### Counts before vi commands

Numbers may be typed as a prefix to some commands. They are interpreted in one of these ways:

<table>
<thead>
<tr>
<th>Count Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>line/column number</td>
<td>z G 1</td>
</tr>
<tr>
<td>scroll amount</td>
<td>^D ^U</td>
</tr>
<tr>
<td>repeat effect</td>
<td>most of the rest</td>
</tr>
</tbody>
</table>

### Interrupting, canceling

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESC</td>
<td>end insert or incomplete command</td>
</tr>
<tr>
<td>DEL</td>
<td>(delete or rubout) interrupts</td>
</tr>
</tbody>
</table>

### File manipulation

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZZ</td>
<td>if file modified, write and exit; otherwise, exit</td>
</tr>
<tr>
<td>:wCR</td>
<td>write back changes</td>
</tr>
<tr>
<td>:w!CR</td>
<td>forced write, if permission originally not valid</td>
</tr>
<tr>
<td>:qCR</td>
<td>quit</td>
</tr>
<tr>
<td>:q!CR</td>
<td>quit, discard changes</td>
</tr>
<tr>
<td>:e nameCR</td>
<td>edit file name</td>
</tr>
<tr>
<td>:e!CR</td>
<td>reedit, discard changes</td>
</tr>
<tr>
<td>:e + nameCR</td>
<td>edit, starting at end</td>
</tr>
<tr>
<td>:e +nCR</td>
<td>edit, starting at line n</td>
</tr>
<tr>
<td>:e #CR</td>
<td>edit alternate file</td>
</tr>
<tr>
<td>:e! #CR</td>
<td>edit alternate file, discard changes</td>
</tr>
<tr>
<td>:w nameCR</td>
<td>write file name</td>
</tr>
<tr>
<td>:w! nameCR</td>
<td>overwrite file name</td>
</tr>
<tr>
<td>:shCR</td>
<td>run shell, then return</td>
</tr>
<tr>
<td>:cmdCR</td>
<td>run cmd, then return</td>
</tr>
<tr>
<td>:nCR</td>
<td>edit next file in arglist</td>
</tr>
<tr>
<td>:n argsCR</td>
<td>specify new arglist</td>
</tr>
<tr>
<td>^G</td>
<td>show current file and line</td>
</tr>
<tr>
<td>:ta tagCR</td>
<td>position cursor to tag</td>
</tr>
</tbody>
</table>
In general, any ex or ed command (such as substitute or global) may be typed, preceded by a colon and followed by a carriage return.

<table>
<thead>
<tr>
<th>Positioning within file</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>forward screen</td>
</tr>
<tr>
<td>^B</td>
<td>backward screen</td>
</tr>
<tr>
<td>^D</td>
<td>scroll down half screen</td>
</tr>
<tr>
<td>^U</td>
<td>scroll up half screen</td>
</tr>
<tr>
<td>nG</td>
<td>go to the beginning of the specified line (end default), where n is a line number</td>
</tr>
<tr>
<td>/pat</td>
<td>next line matching pat</td>
</tr>
<tr>
<td>?pat</td>
<td>previous line matching pat</td>
</tr>
<tr>
<td>n</td>
<td>repeat last / or ? command</td>
</tr>
<tr>
<td>N</td>
<td>reverse last / or ? command</td>
</tr>
<tr>
<td>/pat/+n</td>
<td>nth line after pat</td>
</tr>
<tr>
<td>?pat?−n</td>
<td>nth line before pat</td>
</tr>
<tr>
<td>]</td>
<td>next section/function</td>
</tr>
<tr>
<td>]</td>
<td>previous section/function</td>
</tr>
<tr>
<td>(</td>
<td>beginning of sentence</td>
</tr>
<tr>
<td>)</td>
<td>end of sentence</td>
</tr>
<tr>
<td>{</td>
<td>beginning of paragraph</td>
</tr>
<tr>
<td>}</td>
<td>end of paragraph</td>
</tr>
<tr>
<td>%</td>
<td>find matching ( ) or { }</td>
</tr>
<tr>
<td>^L</td>
<td>clear and redraw window</td>
</tr>
<tr>
<td>^R</td>
<td>clear and redraw window if ^L is → key</td>
</tr>
<tr>
<td>z.CR</td>
<td>redraw screen with current line at top of window</td>
</tr>
<tr>
<td>z−CR</td>
<td>redraw screen with current line at bottom of window</td>
</tr>
<tr>
<td>z.CR</td>
<td>redraw screen with current line at center of window</td>
</tr>
<tr>
<td>/pat/z−CR</td>
<td>move pat line to bottom of window</td>
</tr>
<tr>
<td>zn.CR</td>
<td>use n−line window</td>
</tr>
<tr>
<td>^E</td>
<td>scroll window down one line</td>
</tr>
<tr>
<td>^Y</td>
<td>scroll window up one line</td>
</tr>
<tr>
<td>′′</td>
<td>move cursor to previous context</td>
</tr>
<tr>
<td>′′′</td>
<td>move cursor to first non-white space in line</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Adjusting the screen</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>z.CR</td>
<td>redraw screen with current line at top of window</td>
</tr>
<tr>
<td>z−CR</td>
<td>redraw screen with current line at bottom of window</td>
</tr>
<tr>
<td>z.CR</td>
<td>redraw screen with current line at center of window</td>
</tr>
<tr>
<td>/pat/z−CR</td>
<td>move pat line to bottom of window</td>
</tr>
<tr>
<td>zn.CR</td>
<td>use n−line window</td>
</tr>
<tr>
<td>^E</td>
<td>scroll window down one line</td>
</tr>
<tr>
<td>^Y</td>
<td>scroll window up one line</td>
</tr>
<tr>
<td>′′</td>
<td>move cursor to previous context</td>
</tr>
<tr>
<td>′′′</td>
<td>move cursor to first non-white space in line</td>
</tr>
</tbody>
</table>
vi(1)

<table>
<thead>
<tr>
<th>Character positionning</th>
<th>Line positioning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>x</code></td>
<td>mark current position with the ASCII lower-case letter x</td>
</tr>
<tr>
<td>\x</td>
<td>move cursor to mark x</td>
</tr>
<tr>
<td><code>x</code></td>
<td>move cursor to first non-white space in line marked by x</td>
</tr>
<tr>
<td>H</td>
<td>top line on screen</td>
</tr>
<tr>
<td>L</td>
<td>last line on screen</td>
</tr>
<tr>
<td>M</td>
<td>middle line on screen</td>
</tr>
<tr>
<td>+</td>
<td>next line, at first non-white space character</td>
</tr>
<tr>
<td>−</td>
<td>previous line, at first non-white space character</td>
</tr>
<tr>
<td>CR</td>
<td>return, same as +</td>
</tr>
<tr>
<td>down-arrow</td>
<td>next line, same column</td>
</tr>
<tr>
<td>or j</td>
<td></td>
</tr>
<tr>
<td>up-arrow</td>
<td>previous line, same column</td>
</tr>
<tr>
<td>or k</td>
<td></td>
</tr>
<tr>
<td>^</td>
<td>first non-white space character</td>
</tr>
<tr>
<td>0</td>
<td>beginning of line</td>
</tr>
<tr>
<td>$</td>
<td>end of line</td>
</tr>
<tr>
<td>1 or →</td>
<td>forward</td>
</tr>
<tr>
<td>h or ←</td>
<td>backward</td>
</tr>
<tr>
<td>^H</td>
<td>same as ← (backspace)</td>
</tr>
<tr>
<td>space</td>
<td>same as → (space bar)</td>
</tr>
<tr>
<td>f&lt;sub&gt;x&lt;/sub&gt;</td>
<td>find next x</td>
</tr>
<tr>
<td>F&lt;sub&gt;x&lt;/sub&gt;</td>
<td>find previous x</td>
</tr>
<tr>
<td>t&lt;sub&gt;x&lt;/sub&gt;</td>
<td>move to character following the next x</td>
</tr>
<tr>
<td>T&lt;sub&gt;x&lt;/sub&gt;</td>
<td>move to character following the previous x</td>
</tr>
<tr>
<td>;</td>
<td>repeat last f, F, t, or T</td>
</tr>
<tr>
<td>,</td>
<td>repeat inverse of last f, F, t, or T</td>
</tr>
<tr>
<td>n</td>
<td></td>
</tr>
<tr>
<td>%</td>
<td>find matching ( ) or { }</td>
</tr>
<tr>
<td>w</td>
<td>forward a word</td>
</tr>
<tr>
<td>b</td>
<td>back a word</td>
</tr>
<tr>
<td>e</td>
<td>end of word</td>
</tr>
<tr>
<td>)</td>
<td>to next sentence</td>
</tr>
<tr>
<td>}</td>
<td>to next paragraph</td>
</tr>
<tr>
<td>(</td>
<td>back a sentence</td>
</tr>
<tr>
<td>{</td>
<td>back a paragraph</td>
</tr>
<tr>
<td>W</td>
<td>forward a blank-delimited word</td>
</tr>
<tr>
<td>B</td>
<td>back a blank-delimited word</td>
</tr>
<tr>
<td>E</td>
<td>end of a blank-delimited word</td>
</tr>
<tr>
<td>^H</td>
<td>erase last character (backspace)</td>
</tr>
<tr>
<td>^W</td>
<td>erase last word</td>
</tr>
<tr>
<td>erase</td>
<td>your erase character, same as ^H (backspace)</td>
</tr>
<tr>
<td>kill</td>
<td>your kill character, erase this line of input</td>
</tr>
<tr>
<td>\</td>
<td>quotes your erase and kill characters</td>
</tr>
<tr>
<td>ESC</td>
<td>ends insertion, back to command mode</td>
</tr>
<tr>
<td>CTRL—C</td>
<td>interrupt, suspends insert mode</td>
</tr>
<tr>
<td>^D</td>
<td>backtab one character; reset left margin of autoindent</td>
</tr>
<tr>
<td>^D^D</td>
<td>caret (^) followed by control-d (^D); backtab to beginning of line; do not reset left margin of autoindent</td>
</tr>
<tr>
<td>0^D</td>
<td>backtab to beginning of line; reset left margin of autoindent</td>
</tr>
<tr>
<td>^V</td>
<td>quote non-printable character</td>
</tr>
<tr>
<td>a</td>
<td>append after cursor</td>
</tr>
<tr>
<td>A</td>
<td>append at end of line</td>
</tr>
<tr>
<td>i</td>
<td>insert before cursor</td>
</tr>
<tr>
<td>I</td>
<td>insert before first non-blank</td>
</tr>
<tr>
<td>o</td>
<td>open line below</td>
</tr>
<tr>
<td>O</td>
<td>open line above</td>
</tr>
<tr>
<td>rx</td>
<td>replace single character with x</td>
</tr>
<tr>
<td>R</td>
<td>textESC</td>
</tr>
</tbody>
</table>

**Insert and replace**

**Operators**

Operators are followed by a cursor motion and affect all text that would have been moved over. For example, since w moves over a word, dw deletes the word that would be moved over. Double the operator, for example dd, to affect whole lines.

| d | delete |
| c | change |
| y | yank lines to buffer |
### Miscellaneous Operations

- `<` left shift
- `>` right shift
- `!` filter through command
- `C` change rest of line (c$)
- `D` delete rest of line (d$)
- `s` substitute characters (c1)
- `S` substitute lines (cc)
- `J` join lines
- `x` delete characters (d1)
- `X` delete characters before cursor (dh)
- `Y` yank lines (yy)

### Yank and Put

Put inserts the text most recently deleted or yanked; however, if a buffer is named (using the ASCII lower-case letters a - z), the text in that buffer is put instead.

- `3yy` yank 3 lines
- `3yl` yank 3 characters
- `P` put back text after cursor
- `P` put back text before cursor
- "xp" put from buffer x
- "xy" yank to buffer x
- "xd" delete into buffer x

### Undo, Redo, Retrieve

- `u` undo last change
- `U` restore current line
- . repeat last change
- "dp" retrieve d'th last delete

### USAGE

See largefile(5) for the description of the behavior of vi and view when encountering files greater than or equal to 2 Gbyte ($2^{31}$ bytes).

### ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of vi: LC_CTYPE, LC_TIME, LC_MESSAGES, and NLSPATH.

### FILES

- `/var/tmp` default directory where temporary work files are placed; it can be changed using the directory option (see the ex(1) command)
- `/usr/share/lib/terminfo/?/*` compiled terminal description database
ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>Not enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWxcu4</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

SEE ALSO

intro(1), ctags(1), ed(1), edit(1), ex(1), attributes(5), environ(5), largefile(5), standards(5)

Solaris Advanced User’s Guide

AUTHOR

vi and ex were developed by The University of California, Berkeley California, Computer Science Division, Department of Electrical Engineering and Computer Science.

NOTES

Two options, although they continue to be supported, have been replaced in the documentation by options that follow the Command Syntax Standard (see intro(1)).

An -r option that is not followed with an option-argument has been replaced by -L and +command has been replaced by -c command.

The message file too large to recover with -r option, which is seen when a file is loaded, indicates that the file can be edited and saved successfully, but if the editing session is lost, recovery of the file with the -r option will not be possible.

The editing environment defaults to certain configuration options. When an editing session is initiated, vi attempts to read the EXINIT environment variable. If it exists, the editor uses the values defined in EXINIT; otherwise the values set in $HOME/.exrc are used. If $HOME/.exrc does not exist, the default values are used.

To use a copy of .exrc located in the current directory other than $HOME, set the exrc option in EXINIT or $HOME/.exrc. Options set in EXINIT can be turned off in a local .exrc only if exrc is set in EXINIT or $HOME/.exrc.

Tampering with entries in /usr/share/lib/terminfo/?/* or /usr/share/lib/terminfo/?/* (for example, changing or removing an entry) can affect programs such as vi that expect the entry to be present and correct. In particular, removing the “dumb” terminal may cause unexpected problems.
Software tabs using ^T work only immediately after the autoindent.

Left and right shifts on intelligent terminals do not make use of insert and delete character operations in the terminal.

The standard Solaris version of vi will be replaced by the POSIX.2-conforming version (see standards(5)) in the future. Scripts which use the ex family of addressing and features should use the /usr/xpg4/bin version of these utilities.
NAME  
vipw – edit the password file

SYNOPSIS  
/usr/ucb/vipw

DESCRIPTION  
vipw edits the password file while setting the appropriate locks, and does any necessary processing after the password file is unlocked. If the password file is already being edited, then you will be told to try again later. The vi(1) editor will be used unless the environment variable VISUAL or EDITOR indicates an alternate editor.

vipw performs a number of consistency checks on the password entry for root, and will not allow a password file with a "mangled" root entry to be installed. It also checks the /etc/shells file to verify the login shell for root.

FILES  
/etc/ptmp
/etc/shells

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

SEE ALSO  
passwd(1), vi(1), passwd(4), attributes(5)
volcancel(1)

NAME  volcancel – cancel user’s request for removable media that is not currently in drive

SYNOPSIS  /usr/lib/vold/volcancel [-n] [volume]

DESCRIPTION  volcancel cancels a user’s request to access a particular floppy or CD-ROM file
system. This command is useful when the removable media containing the file system
is not currently in the drive.

Use the path /vol/rdsk/name_of_volume to specify the volume. If called without a
volume name to cancel, volcancel checks for Volume Management running.

OPTIONS  -n  Display the nickname to the device name translation table.

EXAMPLES  EXAMPLE 1 A sample of the volcancel command.

To cancel a request to access an unnamed CD-ROM, use
example$ /usr/lib/vold/volcancel vol/rdsk/unnamed_cdrom

To check if volume management is running, use:
example$ /usr/lib/vold/volcancel | | echo volmgmt not running

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWvolu</td>
</tr>
</tbody>
</table>

SEE ALSO  rmmount(1M), volcheck(1), vold(1M), volmissing(1), rmmount.conf(4),
vold.conf(4), attributes(5), volfs(7FS)
NAME
volcheck – checks for media in a drive and by default checks all floppy media

SYNOPSIS
volcheck [-v] [-i secs] [-t secs] pathname

DESCRIPTION
The volcheck utility tells Volume Management to look at each dev/pathname in sequence and determine if new media has been inserted in the drive.

The default action is to volcheck all checkable media managed by volume management.

OPTIONS
The following options are supported:
- -i secs  Set the frequency of device checking to secs seconds. The default is 2 seconds. The minimum frequency is 1 second.
- -t secs  Check the named device(s) for the next secs seconds. The maximum number of seconds allowed is 28800, which is 8 hours. The frequency of checking is specified by -i. There is no default total time.
- -v  Verbose.

OPERANDS
The following operands are supported:
pathname  The path name of a media device.

EXAMPLES
EXAMPLE 1  A sample of the volcheck command.

The following example

```
example% volcheck -v /dev/diskette
/dev/diskette has media
```

asks Volume Management to examine the floppy drive for new media.

The following example

```
example% volcheck -i 2 -t 600 /dev/diskette &
```

asks Volume Management if there is a floppy in the floppy drive every 2 seconds for 600 seconds (10 minutes).

FILES
/dev/volctl  Volume Management control port

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWvolu</td>
</tr>
</tbody>
</table>

SEE ALSO
eject(1), volcancel(1), volmissing(1), rmmount(1M), vold(1M), rmmount.conf(4), vold.conf(4), attributes(5), volfs(7FS)
Due to a hardware limitation in many floppy drives, the act of checking for media causes mechanical action in the floppy drive. Continuous polling of the floppy drive will cause the drive to wear out. It is recommended that polling the drive only be performed during periods of high use.
volmissing(1)

NAME
volmissing – notify user that volume requested is not in the CD-ROM or floppy drive

SYNOPSIS
/usr/lib/vold/volmissing [-c] [-p] [-s] [-m alias]

DESCRIPTION
volmissing informs a user when a requested volume is not available. Depending on
the option selected, users are notified through their console window, syslogd(1M), or
a mail message.

volmissing -p is the default action taken by vold(1M), the Volume Management
daemon, when it needs to notify a user that the requested volume is not available. If
you want to change this default event, modify the /etc/vold.conf file. See
vold.conf(4).

You can change the notification method for your system by editing the vold.conf
configuration file and providing a new option for volmissing in the notify entry
under the Events category.

OPTIONS
-c Send a message to the user's console requesting the volume be
inserted. To end the notification without inserting the requested
volume, use volcancel(I).

-p All volmissing events will be handled through a GUI, provided
a window system is running on the console. If this option is
specified, and no window system is running, all messages go to
the system console.

-s Send one message to the syslogd(1M).

-m alias Send a mail message to the specified mail alias about the missing
volume.

FILES
/etc/vold.conf
Volume Management daemon configuration file. Directs the Volume Management
daemon to control certain devices, and causes action to be taken when specific
criteria is met.

/usr/lib/vold/volmissing_popup
Pop-up used when the -p option is supplied and a window system is running.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWvolu</td>
</tr>
</tbody>
</table>

SEE ALSO
volcancel(I), volcheck(I), rmmount(1M), syslogd(1M), vold(1M),
rmmount.conf(4), vold.conf(4), attributes(5), volfs(7FS)
volrmmount(1)

NAME
volrmmount – call rmmount to mount or unmount media

SYNOPSIS
volrmmount [-i | -e] [name | nickname]
volrmmount [-d]

DESCRIPTION
The volrmmount utility calls rmmount(1M) to, in effect, simulate an insertion (-i) or an ejection (-e). Simulating an insertion often means that rmmount will mount the media. Conversely, simulating an ejection often means that rmmount will unmount the media. However, these actions can vary depending on the rmmount configuration and media type (see rmmount.conf(4)).

For example, if you use the default /etc/rmmount.conf and insert a music CD, it will not be mounted. However, you can configure rmmount so that it calls workman whenever a music CD is inserted.

This command allows you to override Volume Management’s usual handling of media (see EXAMPLES below).

OPTIONS
The following options are supported:
- i Simulates an insertion of the specified media by calling rmmount.
- e Simulates an ejection of the specified media by calling rmmount.
- d Displays the name of the default device for volrmmount to handle. This device is used if no name or nickname is supplied.

OPERANDS
The following operands are supported:
name The name that Volume Management recognizes as the device’s name. See volfs(7FS).
nickname A shortened version of the device’s name. Following is the list of recognized nicknames:

<table>
<thead>
<tr>
<th>Nickname</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>fd</td>
<td>/dev/rdiskette</td>
</tr>
<tr>
<td>fd0</td>
<td>/dev/rdiskette</td>
</tr>
<tr>
<td>fd1</td>
<td>/dev/rdiskette0</td>
</tr>
<tr>
<td>diskette</td>
<td>/dev/rdiskette</td>
</tr>
<tr>
<td>diskette0</td>
<td>/dev/rdiskette0</td>
</tr>
<tr>
<td>diskette1</td>
<td>/dev/rdiskette1</td>
</tr>
<tr>
<td>rdiskette</td>
<td>/dev/rdiskette</td>
</tr>
<tr>
<td>rdiskette0</td>
<td>/dev/rdiskette0</td>
</tr>
</tbody>
</table>
### EXAMPLES

**EXAMPLE 1 Using the volrmmount command**

When Volume Management finds a floppy that contains a filesystem, it calls rmount to mount it. If you wish to run `tar(1)` or `cpio(1)` on that floppy, it must first be unmounted. To unmount the floppy use:

```
example% volrmmount -e floppy0
```

After `volrmmount` unmounts the floppy, if you wish to re-mount it (rather than ejecting it and reinserting it) use:

```
example% volrmmount -i floppy0
```

Notice that if you are using a named floppy, you can use its name in place of floppy0.

### FILES

`/dev/volctl` Volume Management control port

### ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWvolu</td>
</tr>
</tbody>
</table>

### SEE ALSO

`cpio(1), eject(1), tar(1), rmount(1M), vold(1M), rmount.conf(4), attributes(5), volfs(7FS)`

### NOTES

Volume Management (vold) must be running to use this command.
vsig(1F)

NAME      vsig – synchronize a co-process with the controlling FMLI application
SYNOPSIS  vsig

DESCRIPTION The vsig executable sends a SIGUSR2 signal to the controlling FMLI process. This
           signal/alarm causes FMLI to execute the FMLI built-in command checkworld which
           causes all posted objects with a reread descriptor evaluating to TRUE to be reread.
           vsig takes no arguments.

EXAMPLES  EXAMPLE 1 A sample output of vsig command.
           The following is a segment of a shell program:
           echo "Sending this string to an FMLI process"
           vsig
           The vsig executable will flush the output buffer before it sends the SIGUSR2 signal to
           make sure the string is actually in the pipe created by the cocreate function.

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

+---------------------------------+-----------------
| ATTRIBUTE TYPE                  | ATTRIBUTE VALUE |
+---------------------------------+-----------------|
| Availability                   | SUNWesu         |

SEE ALSO  coproc(1F), kill(1), kill(2), signal(3C), attributes(5)

NOTES     Because vsig synchronize with FMLI, it should be used rather than kill to send a
           SIGUSR2 signal to FMLI.
NAME  
w – display information about currently logged-in users

SYNOPSIS  
w [-hlsuw] [user]

DESCRIPTION  
The w command displays a summary of the current activity on the system, including what each user is doing. The heading line shows the current time, the length of time the system has been up, the number of users logged into the system, and the average number of jobs in the run queue over the last 1, 5 and 15 minutes.

The fields displayed are: the user’s login name, the name of the tty the user is on, the time of day the user logged on (in hours:minutes), the idle time—that is, the number of minutes since the user last typed anything (in hours:minutes), the CPU time used by all processes and their children on that terminal (in minutes:seconds), the CPU time used by the currently active processes (in minutes:seconds), and the name and arguments of the current process.

OPTIONS  
The following options are supported:

  -h  Suppresses the heading.
  -l  Produces a long form of output, which is the default.
  -s  Produces a short form of output. In the short form, the tty is abbreviated, the login time and CPU times are left off, as are the arguments to commands.
  -u  Produces the heading line which shows the current time, the length of time the system has been up, the number of users logged into the system, and the average number of jobs in the run queue over the last 1, 5 and 15 minutes.
  -w  Produces a long form of output, which is also the same as the default.

OPERANDS  
user  Name of a particular user for whom login information is displayed. If specified, output is restricted to that user.

EXAMPLES  
EXAMPLE 1  Sample Output From the w Command

example% w
10:54am up 27 day(s), 57 mins, 1 user, load average: 0.28, 0.26, 0.22
User tty login idle JCPU PCPU what
ralph console 7:10am 1 10:05 4:31 w

example% w
10:54am up 27 day(s), 57 mins, 1 user, load average: 0.28, 0.26, 0.22
User tty login idle JCPU PCPU what
ralph console 7:10am 1 10:05 4:31 w

ENVIRONMENT VARIABLES  
See environ(5) for descriptions of the following environment variables that affect the execution of w: LC_CTYPE, LC_MESSAGES, and LC_TIME.

FILES  
/var/adm/utmpx user and accounting information
w(1)

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

**SEE ALSO**

`ps(1), who(1), whodo(1M), utmpx(4), attributes(5), environ(5)`

**NOTES**

The notion of the “current process” is unclear. The current algorithm is ‘the highest numbered process on the terminal that is not ignoring interrupts, or, if there is none, the highest numbered process on the terminal’. This fails, for example, in critical sections of programs like the shell and editor, or when faulty programs running in the background fork and fail to ignore interrupts. In cases where no process can be found, `w` prints `-`

The CPU time is only an estimate, in particular, if someone leaves a background process running after logging out, the person currently on that terminal is “charged” with the time.

Background processes are not shown, even though they account for much of the load on the system.

Sometimes processes, typically those in the background, are printed with null or garbaged arguments. In these cases, the name of the command is printed in parentheses.

`w` does not know about the conventions for detecting background jobs. It will sometimes find a background job instead of the right one.
wait – wait process completion

wait [pid...]

wait [pid...]

wait [% jobid...]

The shell itself executes wait, without creating a new process. If you get the error message cannot fork, too many processes, try using the wait command to clean up your background processes. If this doesn’t help, the system process table is probably full or you have too many active foreground processes. (There is a limit to the number of process IDs associated with your login, and to the number the system can keep track of.)

Not all the processes of a pipeline with three or more stages are children of the shell, and thus cannot be waited for.

Wait for your background process whose process ID is pid and report its termination status. If pid is omitted, all your shell’s currently active background processes are waited for and the return code will be 0. The wait utility accepts a job identifier, when Job Control is enabled (jsh), and the argument, jobid, is preceded by a percent sign (%).

If pid is not an active process ID, the wait utility will return immediately and the return code will be 0.

Wait for your background processes.

When an asynchronous list is started by the shell, the process ID of the last command in each element of the asynchronous list becomes known in the current shell execution environment.

If the wait utility is invoked with no operands, it will wait until all process IDs known to the invoking shell have terminated and exit with an exit status of 0.

If one or more pid or jobid operands are specified that represent known process IDs (or jobids), the wait utility will wait until all of them have terminated. If one or more pid or jobid operands are specified that represent unknown process IDs (or jobids), wait will treat them as if they were known process IDs (or jobids) that exited with exit status 127. The exit status returned by the wait utility will be the exit status of the process requested by the last pid or jobid operand.

The known process IDs are applicable only for invocations of wait in the current shell execution environment.

The following operands are supported:

One of the following:
wait(1)

pid  The unsigned decimal integer process ID of a command, for which the utility is to wait for the termination.

jobid  A job control job ID that identifies a background process group to be waited for. The job control job ID notation is applicable only for invocations of wait in the current shell execution environment, and only on systems supporting the job control option.

**USAGE**

On most implementations, wait is a shell built-in. If it is called in a subshell or separate utility execution environment, such as one of the following,

```
(wait)
nohup wait . . .
find . -exec wait . . . \; it will return immediately because there will be no known process IDs to wait for in those environments.
```

**EXAMPLES**

**EXAMPLE 1 Using A Script To Identify The Termination Signal**

For Although the exact value used when a process is terminated by a signal is unspecified, if it is known that a signal terminated a process, a script can still reliably figure out which signal is using kill, as shown by the following (/bin/ksh and /usr/xpg4/bin/sh):

```
sleep 1000&
pid=$!
kill -kill $pid
wait $pid
```

```
echo $pid was terminated by a SIG$(kill -l $(($?−128))) signal.
```

**EXAMPLE 2 Returning The Exit Status Of A Process**

If the following sequence of commands is run in less than 31 seconds (/bin/ksh and /usr/xpg4/bin/sh):

```
sleep 257 | sleep 31 &
jobs -l %%
```

then either of the following commands will return the exit status of the second sleep in the pipeline:

```
wait <pid of sleep 31>
wait %%
```

**ENVIRONMENT VARIABLES**

See environ(5) for descriptions of the following environment variables that affect the execution of wait: LC_CTYPE, LC_MESSAGES, and NLSPATH.

**ATTRIBUTES**

See attributes(5) for descriptions of the following attributes:
wait(1)

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO  csh(1), jobs(1), ksh(1), sh(1), attributes(5), environ(5)
NAME
wc – display a count of lines, words and characters in a file

SYNOPSIS
wc [-c | -m | -C] [-lwc] [file...]

DESCRIPTION
The wc utility reads one or more input files and, by default, writes the number of newline characters, words and bytes contained in each input file to the standard output.

The utility also writes a total count for all named files, if more than one input file is specified.

wc considers a word to be a non-zero-length string of characters delimited by white space (for example, SPACE, TAB). See iswspace(3C) or isspace(3C).

OPTIONS
The following options are supported:
- c       Count bytes.
- m       Count characters.
- C       Same as -m.
- l       Count lines.
- w       Count words delimited by white space characters or new line characters. Delimiting characters are Extended Unix Code (EUC) characters from any code set defined by iswspace().

If no option is specified the default is -lwc (count lines, words, and bytes.)

OPERANDS
The following operand is supported:
file       A path name of an input file. If no file operands are specified, the standard input will be used.

USAGE
See largefile(5) for the description of the behavior of wc when encountering files greater than or equal to 2 Gbyte (2^31 bytes).

ENVIRONMENT
VARIABLES
See environ(5) for descriptions of the following environment variables that affect the execution of wc: LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS
The following exit values are returned:
0       Successful completion.
>0      An error occurred.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>
### SEE ALSO

- `cksum(1)`, `isspace(3C)`, `iswalpha(3C)`, `iswspace(3C)`, `setlocale(3C)`, `attributes(5)`, `environ(5)`, `largefile(5)`
what(1)

NAME
what – extract SCCS version information from a file

SYNOPSIS
what [-s] filename...

DESCRIPTION
what searches each filename for occurrences of the pattern @(#) that the SCCS get
command (see sccs-get(1)) substitutes for the @(#) ID keyword, and prints what
follows up to a ",", >, NEWLINE, \, or null character.

OPTIONS
The following option is supported:
-s Stop after the first occurrence of the pattern.

EXAMPLES
EXAMPLE 1 Extracting SCCS version information
If a C program in file program.c contains

char scssid[ ] = " @(#)identification information ";

and program.c is compiled to yield program.o and a.out, the command:

example% what program.c program.o a.out

produces:

program.c: identification information
program.o: identification information
a.out: identification information

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWsprot</td>
</tr>
</tbody>
</table>

SEE ALSO
sccs(1), sccs-admin(1), sccs-cdc(1), sccs-comb(1), sccs-delta(1),
sccs-get(1), sccs-help(1), sccs-prs(1), sccs-prt(1), sccs-rmdel(1),
sccs-sact(1), sccs-sccsdiff(1), sccs-unget(1), sccs-val(1), sccsfile(4),
attributes(5)

DIAGNOSTICS
Use the SCCS help command for explanations (see sccs-help(1)).

BUGS
There is a remote possibility that a spurious occurrence of the ‘@(#)’ pattern could be
found by what.
whatis(1)

NAME  whatis – display a one-line summary about a keyword

SYNOPSIS  whatis command...

DESCRIPTION  whatis looks up a given command and displays the header line from the manual section. You can then run the man(1) command to get more information. If the line starts ‘name(section) …’ you can do ‘man -s section name’ to get the documentation for it. Try ‘whatis ed’ and then you should do ‘man -s 1 ed’ to get the manual page for ed(1).

whatis is actually just the -f option to the man(1) command.

whatis uses the /usr/share/man/windex database. This database is created by catman(1M). If this database does not exist, whatis will fail.

FILES  /usr/share/man/windex Table of contents and keyword database

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWdoc</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

SEE ALSO  apropos(1), man(1), catman(1M), attributes(5)
NAME
whereis – locate the binary, source, and manual page files for a command

SYNOPSIS
/usr/ucb/whereis [-bmsu] [-BMS directory... -f] filename...

DESCRIPTION
The `whereis` utility locates source/binary and manuals sections for specified files. The supplied names are first stripped of leading pathname components and any (single) trailing extension of the form `.ext`, for example, `.c`. Prefixes of `.c` resulting from use of source code control are also dealt with. `whereis` then attempts to locate the desired program in a list of standard places:

```
/etc
/sbin
/usr/bin
/usr/ccs/bin
/usr/ccs/lib
/usr/lang
/usr/lib
/usr/sbin
/usr/ucb
/usr/ucblib
/usr/ucbinclude
/usr/games
/usr/local
/usr/local/bin
/usr/new
/usr/old
/usr/hosts
/usr/include
/usr/etc
```

OPTIONS
The following options are supported:

- `-b` Searches only for binaries.
- `-B` Changes or otherwise limits the places where `whereis` searches for binaries.
- `-f` Terminates the last directory list and signals the start of file names, and `must` be used when any of the `-B`, `-M`, or `-S` options are used.
- `-m` Searches only for manual sections.
- `-M` Changes or otherwise limits the places where `whereis` searches for manual sections.
- `-s` Searches only for sources.
- `-S` Changes or otherwise limit the places where `whereis` searches for sources.
- `-u` Searches for unusual entries. A file is said to be unusual if it does not have one entry of each requested type. Thus `whereis -m -u *` asks for those files in the current directory which have no documentation.
EXAMPLES

EXAMPLE 1 Finding files

Find all files in /usr/bin which are not documented in /usr/share/man/man1 with source in /usr/src/cmd:

```bash
eexample cd /usr/ucb
eexample whereis -u -M /usr/share/man/man1 -S /usr/src/cmd -f *
```

FILES

/usr/src/*

/usr/{doc,man}/*

/etc, /usr/{lib,bin,ucb,old,new,local}

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

SEE ALSO

chdir(2), attributes(5)

BUGS

Since whereis uses chdir(2) to run faster, pathnames given with the -M, -S, or -B must be full; that is, they must begin with a `/`. 
which(1)

NAME 
which – locate a command; display its pathname or alias

SYNOPSIS 
which [filename...]

DESCRIPTION 
which takes a list of names and looks for the files which would be executed had these names been given as commands. Each argument is expanded if it is aliased, and searched for along the user’s path. Both aliases and path are taken from the user’s .cshrc file.

FILES
-/ .cshrc 
source of aliases and path values

/usr/bin/which

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO

csh(1), attributes(5)

DIAGNOSTICS
A diagnostic is given for names which are aliased to more than a single word, or if an executable file with the argument name was not found in the path.

NOTES

which is not a shell built-in command; it is the UNIX command, /usr/bin/which

BUGS

Only aliases and paths from –/.cshrc are used; importing from the current environment is not attempted. Must be executed by csh(1), since only csh knows about aliases.

To compensate for –/.cshrc files in which aliases depend upon the prompt variable being set, which sets this variable to NULL. If the –/.cshrc produces output or prompts for input when prompt is set, which may produce some strange results.
**NAME**

who – who is on the system

**SYNOPSIS**

```
/usr/bin/who [-abdHlmpqrstTu] [file]
/usr/bin/who -q [-n x] [file]
/usr/bin/who am i
/usr/bin/who am I
/usr/xpg4/bin/who [-abdHlmpqrtu] [file]
/usr/xpg4/bin/who -q [-n x] [file]
/usr/xpg4/bin/who -s [-bdHlmpqrtu] [file]
/usr/xpg4/bin/who am i
/usr/xpg4/bin/who am I
```

**DESCRIPTION**

The `who` utility can list the user’s name, terminal line, login time, elapsed time since activity occurred on the line, and the process-ID of the command interpreter (shell) for each current UNIX system user. It examines the `/var/adm/utmpx` file to obtain its information. If `file` is given, that file (which must be in `utmpx(4)` format) is examined. Usually, `file` will be `/var/adm/wtmpx`, which contains a history of all the logins since the file was last created.

The general format for output is:

```
name [state] line time [idle] [pid] [comment] [exit]
```

where:

- **name**: User’s login name
- **state**: Capability of writing to the terminal
- **line**: Name of the line found in `/dev`
- **time**: Time since user’s login
- **idle**: Time elapsed since the user’s last activity
- **pid**: User’s process id
- **comment**: Comment line in `inittab(4)`
- **exit**: Exit status for dead processes

**OPTIONS**

The following options are supported:

- `-a` Processes `/var/adm/utmpx` or the named `file` with `-b`, `-d`, `-l`, `-p`, `-r`, `-t`, `-T`, and `-u` options turned on.
- `-b` Indicates the time and date of the last reboot.
who(1)

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-d</td>
<td>Displays all processes that have expired and not been respawned by <code>init</code>. The <code>exit</code> field appears for dead processes and contains the termination and exit values (as returned by <code>wait(3UCB)</code>), of the dead process. This can be useful in determining why a process terminated.</td>
</tr>
<tr>
<td>-H</td>
<td>Outputs column headings above the regular output.</td>
</tr>
<tr>
<td>-l</td>
<td>Lists only those lines on which the system is waiting for someone to login. The <code>name</code> field is <code>LOGIN</code> in such cases. Other fields are the same as for user entries except that the <code>state</code> field does not exist.</td>
</tr>
<tr>
<td>-m</td>
<td>Outputs only information about the current terminal.</td>
</tr>
<tr>
<td>-n x</td>
<td>Takes a numeric argument, <code>x</code>, which specifies the number of users to display per line. <code>x</code> must be at least 1. The <code>-n</code> option can only be used with <code>-q</code>.</td>
</tr>
<tr>
<td>-p</td>
<td>Lists any other process that is currently active and has been previously spawned by <code>init</code>. The <code>name</code> field is the name of the program executed by <code>init</code> as found in <code>/sbin/inittab</code>. The <code>state</code>, <code>line</code>, and <code>idle</code> fields have no meaning. The <code>comment</code> field shows the <code>id</code> field of the line from <code>/sbin/inittab</code> that spawned this process. See <code>inittab(4)</code>.</td>
</tr>
<tr>
<td>-q</td>
<td>(Quick who) Displays only the names and the number of users currently logged on. When this option is used, all other options are ignored.</td>
</tr>
<tr>
<td>-r</td>
<td>Indicates the current run-level of the <code>init</code> process.</td>
</tr>
<tr>
<td>-s</td>
<td>(Default) Lists only the <code>name</code>, <code>line</code>, and <code>time</code> fields.</td>
</tr>
</tbody>
</table>

**/usr/bin/who**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-T</td>
<td>Same as the <code>-s</code> option, except that the <code>state idle</code>, <code>pid</code>, and <code>comment</code> fields are also written. <code>state</code> is one of the following characters:</td>
</tr>
<tr>
<td>+</td>
<td>The terminal allows write access to other users.</td>
</tr>
<tr>
<td>-</td>
<td>The terminal denies write access to other users.</td>
</tr>
<tr>
<td>?</td>
<td>The terminal write-access state cannot be determined.</td>
</tr>
</tbody>
</table>

**/usr/xpg4/bin/who**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-T</td>
<td>Same as the <code>-s</code> option, except that the <code>state</code> field is also written. <code>state</code> is one of the characters listed under the <code>/usr/bin/who</code> version of this option. If the <code>-u</code> option is used with <code>-T</code>, the idle time is added to the end of the previous format.</td>
</tr>
<tr>
<td>-t</td>
<td>Indicates the last change to the system clock (using the <code>date</code> utility) by <code>root</code>. See <code>su(1M)</code> and <code>date(1)</code>.</td>
</tr>
<tr>
<td>-u</td>
<td>Lists only those users who are currently logged in. The <code>name</code> is the user's login name. The <code>line</code> is the name of the line as found in the directory <code>/dev</code>. The <code>time</code> is the time that the user logged in. The <code>idle</code> column contains the number of hours and minutes since activity last occurred on that particular line. A dot (.) indicates that the terminal has seen activity in the last minute and is therefore “current.” If more than twenty-four hours have elapsed or the line has not been used since boot time, the entry is marked</td>
</tr>
</tbody>
</table>
old. This field is useful when trying to determine whether a person is
working at the terminal or not. The pid is the process-ID of the user’s shell.
The comment is the comment field associated with this line as found in
/sbin/inittab (see inittab(4)). This can contain information about
where the terminal is located, the telephone number of the dataset, type of
terminal if hard-wired, and so forth.

OPERANDS

The following operands are supported:

am i
am I In the “C” locale, limits the output to describing the invoking user,
equivalent to the -m option. The am and i or I must be separate
arguments.

file Specifies a path name of a file to substitute for the database of logged-on
users that who uses by default.

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the
execution of who: LC_CTYPE, LC_MESSAGES, LC_TIME, and NLSPATH.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

FILES

/sbin/inittab Script for init
/var/adm/utmpx Current user and accounting information
/var/adm/wtmpx Historic user and accounting information

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>/usr/bin/who</th>
<th>/usr/xpg4/bin/who</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTRIBUTE TYPE</td>
<td>ATTRIBUTE VALUE</td>
</tr>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO

date(1), login(1), msg(1), init(1M), su(1M), wait(3UCB), inittab(4), utmpx(4),
attributes(5), environ(5), XPG4(5)

NOTES

Super user: After a shutdown to the single-user state, who returns a prompt; since
/var/adm/utmpx is updated at login time and there is no login in single-user state,
who cannot report accurately on this state. who am i, however, returns the correct
information.
whoami(1B)

NAME    whoami – display the effective current username

SYNOPSIS  /usr/ucb/whoami

DESCRIPTION  whoami displays the login name corresponding to the current effective user ID. If you have used su to temporarily adopt another user, whoami will report the login name associated with that user ID. whoami gets its information from the geteuid and getpwuid library routines (see getuid and getpwnam(3C), respectively).

FILES    /etc/passwd  username data base

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWscpu</td>
</tr>
</tbody>
</table>

SEE ALSO  su(1M), who(1), getuid(2), getpwnam(3C), attributes(5)
whocalls(1)

NAME
whocalls – report on the calls to a specific procedure

SYNOPSIS
/usr/ccs/bin/whocalls [-l wholib] [-s] funcname executable [arguments...]

DESCRIPTION
whocalls is a simple example of a utility based on the Link-Auditing functionality of ld.so.1(1) that permits the tracking of a given function call. See the Linker and Libraries Guide for a detailed description of the Link-Auditing mechanism. The executable is run as normal with any associated arguments. Each time the procedure funcname is called, both the arguments to that procedure and a stack trace are displayed on standard output.

OPTIONS
The following options are supported:

- `l wholib` Specifies an alternate who.so Link-Auditing library to use.
- `-s` When available, examines and uses the .symtab symbol table for local symbols. This is a little more expensive than using the .dynsym symbol table, but can produce more detailed stack trace information.

EXAMPLES

EXAMPLE 1 Tracking function calls

This example tracks the calls to printf() made by a simple helloworld program:

```
example$ whocalls printf helloworld
printf(0x106e4, 0xef625310, 0xef621ba8)
helloworld::main+0x10
helloworld::_start+0x5c
Hello World
```

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWtoo</td>
</tr>
</tbody>
</table>

SEE ALSO

ld.so.1(1), sotruss(1), attributes(5)

Linker and Libraries Guide
whois

NAME

whois – Internet user name directory service

SYNOPSIS

whois [-h host] identifier

DESCRIPTION

whois searches for an Internet directory entry for an identifier which is either a name (such as “Smith”) or a handle (such as “SRI-NIC”). To force a name-only search, precede the name with a period; to force a handle-only search, precede the handle with an exclamation point.

To search for a group or organization entry, precede the argument with * (an asterisk). The entire membership list of the group will be displayed with the record.

You may of course use an exclamation point and asterisk, or a period and asterisk together.

EXAMPLES

EXAMPLE 1 Using The whois Command

The command:

example$ whois Smith

looks for the name or handle SMITH.

The command:

example$ whois !SRI-NIC

looks for the handle SRI-NIC only.

The command:

example$ whois .Smith, John

looks for the name JOHN SMITH only.

Adding ... to the name or handle argument will match anything from that point; that is, ZU ... will match ZUL, ZUM, and so on.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWrcmdc</td>
</tr>
</tbody>
</table>

SEE ALSO

attributes(5)
write(1)

NAME
write – write to another user

SYNOPSIS
write user [terminal]

DESCRIPTION
The write utility reads lines from the user’s standard input and writes them to the
terminal of another user. When first invoked, it writes the message:
Message from sender-login-id (sending-terminal) [date]...
to user. When it has successfully completed the connection, the sender’s terminal will
be alerted twice to indicate that what the sender is typing is being written to the
recipient’s terminal.

If the recipient wants to reply, this can be accomplished by typing
write sender-login-id [sending-terminal] upon receipt of the initial message. Whenever a line of
input as delimited by a NL, EOF, or EOL special character is accumulated while in
canonical input mode, the accumulated data will be written on the other user’s
terminal. Characters are processed as follows:

- Typing the alert character will write the alert character to the recipient’s terminal.
- Typing the erase and kill characters will affect the sender’s terminal in the manner
described by the termios(3C) interface.
- Typing the interrupt or end-of-file characters will cause write to write an
  appropriate message (EOT\n in the "C" locale) to the recipient’s terminal and exit.
- Typing characters from LC_CTYPE classifications print or space will cause those
  characters to be sent to the recipient’s terminal.
- When and only when the stty iexten local mode is enabled, additional special
  control characters and multi-byte or single-byte characters are processed as
  printable characters if their wide character equivalents are printable.
- Typing other non-printable characters will cause them to be written to the
  recipient’s terminal as follows: control characters will appear as a ‘‘^’’ followed by
  the appropriate ASCII character, and characters with the high-order bit set will
  appear in “meta” notation. For example, ‘‘\003’’ is displayed as ‘‘^C’’ and ‘‘\372’’ as
  ‘‘M−Z’’.

To write to a user who is logged in more than once, the terminal argument can be used
to indicate which terminal to write to; otherwise, the recipient’s terminal is the first
writable instance of the user found in /usr/adm/utmpx, and the following
informational message will be written to the sender’s standard output, indicating
which terminal was chosen:

usr is logged on more than one place.
You are connected to terminal.
Other locations are: terminal

Permission to be a recipient of a write message can be
denied or granted by use of the msg utility. However, a user’s privilege may further
constrain the domain of accessibility of other users’ terminals. The write utility will
fail when the user lacks the appropriate privileges to perform the requested action.
If the character `!` is found at the beginning of a line, `write` calls the shell to execute the rest of the line as a command.

`write` runs `setgid()` (see `setuid(2)`) to the group ID `tty`, in order to have write permissions on other user’s terminals.

The following protocol is suggested for using `write`: when you first `write` to another user, wait for them to `write` back before starting to send. Each person should end a message with a distinctive signal (that is, `(o)` for “over”) so that the other person knows when to reply. The signal `((o))` (for “over and out”) is suggested when conversation is to be terminated.

**OPERANDS**
The following operands are supported:

- **user**
  User (login) name of the person to whom the message will be written. This operand must be of the form returned by the `who(1)` utility.

- **terminal**
  Terminal identification in the same format provided by the `who` utility.

**ENVIRONMENT VARIABLES**
See `environ(5)` for descriptions of the following environment variables that affect the execution of `write`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

**EXIT STATUS**
The following exit values are returned:

- **0**
  Successful completion.

- **>0**
  The addressed user is not logged on or the addressed user denies permission.

**FILES**

- `/var/adm/utmpx` user and accounting information for `write`

- `/usr/bin/sh` Bourne shell executable file

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

**SEE ALSO**

- `mail(1)`, `msg(1)`, `pr(1)`, `sh(1)`, `talk(1)`, `who(1)`, `setuid(2)`, `termios(3C)`, `attributes(5)`, `environ(5)`

**DIAGNOSTICS**

- **user is not logged on**
  The person you are trying to `write` to is not logged on.

- **Permission denied**
  The person you are trying to `write` to denies that permission (with `msg`).
write(1)

Warning: cannot respond, set mesg -y
   Your terminal is set to mesg n and the recipient cannot respond to you.

Can no longer write to user
   The recipient has denied permission (mesg n) after you had started writing.
xargs(1)

NAME  xargs – construct argument lists and invoke utility

SYNOPSIS  xargs [-t] [-p] [-e [eofstr]] [-E eofstr] [-I replstr] [-i [replstr]]
 [-L number] [-l [number]] [-n number] [-x] [-s size] [utility
 [argument. . .]]

DESCRIPTION  The xargs utility constructs a command line consisting of the utility and argument
 operands specified followed by as many arguments read in sequence from standard
 input as will fit in length and number constraints specified by the options. The xargs
 utility then invokes the constructed command line and waits for its completion. This
 sequence is repeated until an end-of-file condition is detected on standard input or an
 invocation of a constructed command line returns an exit status of 255.

Arguments in the standard input must be separated by unquoted blank characters, or
 unescaped blank characters or newline characters. A string of zero or more
 non-double-quote (\") and non-newline characters can be quoted by enclosing them in
 double-quotes. A string of zero or more non-apostrophe (\') and non-newline
 characters can be quoted by enclosing them in apostrophes. Any unquoted character
 can be escaped by preceding it with a backslash (\). The utility will be executed one or
 more times until the end-of-file is reached. The results are unspecified if the utility
 named by utility attempts to read from its standard input.

The generated command line length will be the sum of the size in bytes of the utility
 name and each argument treated as strings, including a null byte terminator for each
 of these strings. The xargs utility will limit the command line length such that when
 the command line is invoked, the combined argument and environment lists will not
 exceed {ARG_MAX}−2048 bytes. Within this constraint, if neither the -n nor the -s
 option is specified, the default command line length will be at least {LINE_MAX}.

OPTIONS  The following options are supported:

- e [eofstr]  Use eofstr as the logical end-of-file string. Underscore (_) is
 assumed for the logical EOF string if neither -e nor -E is used.
 When the -eofstr option-argument is omitted, the logical EOF
 string capability is disabled and underscores are taken literally.
 The xargs utility reads standard input until either end-of-file or
 the logical EOF string is encountered.

- E eofstr  Specify a logical end-of-file string to replace the default
 underscore. The xargs utility reads standard input until either
 end-of-file or the logical EOF string is encountered.

- I replstr  Insert mode. utility will be executed for each line from standard
 input, taking the entire line as a single argument, inserting it in
 argument s for each occurrence of replstr. A maximum of five
 arguments in arguments can each contain one or more instances of
 replstr. Any blank characters at the beginning of each line are
 ignored. Constructed arguments cannot grow larger than 255
 bytes. Option -x is forced on. The -I and -i options are mutually
 exclusive; the last one specified takes effect.
This option is equivalent to `-I replstr`. The string `{ }` is assumed for `replstr` if the option-argument is omitted.

The utility will be executed for each non-empty `number` lines of arguments from standard input. The last invocation of `utility` will be with fewer lines of arguments if fewer than `number` remain. A line is considered to end with the first newline character unless the last character of the line is a blank character; a trailing blank character signals continuation to the next non-empty line, inclusive. The `-L`, `-l`, and `-n` options are mutually exclusive; the last one specified takes effect.

(The letter ell.) This option is equivalent to `-L number`. If `number` is omitted, 1 is assumed. Option `-x` is forced on.

Invoke `utility` using as many standard input arguments as possible, up to `number` (a positive decimal integer) arguments maximum. Fewer arguments will be used if:

- The command line length accumulated exceeds the size specified by the `-s` option (or `{LINE_MAX}` if there is no `-s` option), or
- The last iteration has fewer than `number`, but not zero, operands remaining.

Prompt mode. The user is asked whether to execute `utility` at each invocation. Trace mode (`-t`) is turned on to write the command instance to be executed, followed by a prompt to standard error. An affirmative response (specific to the user’s locale) read from `/dev/tty` will execute the command; otherwise, that particular invocation of `utility` is skipped.

Invoke `utility` using as many standard input arguments as possible yielding a command line length less than `size` (a positive decimal integer) bytes. Fewer arguments will be used if:

- The total number of arguments exceeds that specified by the `-n` option, or
- The total number of lines exceeds that specified by the `-L` option, or
- End of file is encountered on standard input before `size` bytes are accumulated.

Values of `size` up to at least `{LINE_MAX}` bytes are supported, provided that the constraints specified in DESCRIPTION are met. It is not considered an error if a value larger than that supported by the implementation or exceeding the constraints specified in DESCRIPTION is given; `xargs` will use the largest value it supports within the constraints.
Enable trace mode. Each generated command line will be written to standard error just prior to invocation.

Terminate if a command line containing number arguments (see the -n option above) or number lines (see the -L option above) will not fit in the implied or specified size (see the -s option above).

OPERANDS
The following operands are supported:

utility The name of the utility to be invoked, found by search path using the PATH environment variable; see environ(5). If utility is omitted, the default is the echo(1) utility. If the utility operand names any of the special built-in utilities in shell_builtins(1), the results are undefined.

argument An initial option or operand for the invocation of utility.

USAGE
The 255 exit status allows a utility being used by xargs to tell xargs to terminate if it knows no further invocations using the current data stream will succeed. Thus, utility should explicitly exit with an appropriate value to avoid accidentally returning with 255.

Note that input is parsed as lines; blank characters separate arguments. If xargs is used to bundle output of commands like find dir -print or ls into commands to be executed, unexpected results are likely if any filenames contain any blank characters or newline characters. This can be fixed by using find to call a script that converts each file found into a quoted string that is then piped to xargs. Note that the quoting rules used by xargs are not the same as in the shell. They were not made consistent here because existing applications depend on the current rules and the shell syntax is not fully compatible with it. An easy rule that can be used to transform any string into a quoted form that xargs will interpret correctly is to precede each character in the string with a backslash (\).

On implementations with a large value for \{ARG_MAX\}, xargs may produce command lines longer than \{LINE_MAX\}. For invocation of utilities, this is not a problem. If xargs is being used to create a text file, users should explicitly set the maximum command line length with the -s option.

The xargs utility returns exit status 127 if an error occurs so that applications can distinguish “failure to find a utility” from “invoked utility exited with an error indication.” The value 127 was chosen because it is not commonly used for other meanings; most utilities use small values for “normal error conditions” and the values above 128 can be confused with termination due to receipt of a signal. The value 126 was chosen in a similar manner to indicate that the utility could be found, but not invoked.

EXAMPLES

EXAMPLE 1 Using The xargs Command

The following will move all files from directory $1 to directory $2, and echo each move command just before doing it:
EXAMPLE 1 Using The xargs Command (Continued)

ls $1 | xargs -I {} -t mv $1/{} $2/{}

The following command will combine the output of the parenthesised commands onto one line, which is then written to the end of file log:

(logname; date; printf "\%s\n" "$0 $*") | xargs >>log

The following command will invoke diff with successive pairs of arguments originally typed as command line arguments (assuming there are no embedded blank characters in the elements of the original argument list):

printf "%s\n" "$*" | xargs -n 2 -x diff

The user is asked which files in the current directory are to be archived. The files are archived into arch; a, one at a time, or b, many at a time:

ls | xargs -p -L 1 ar -r arch
ls | xargs -p -L 1 | xargs ar -r arch

The following will execute with successive pairs of arguments originally typed as command line arguments:

echo $* | xargs -n 2 diff

See environ(5) for descriptions of the following environment variables that affect the execution of xargs: LC_COLLATE, LC_CTYPE, LC_MESSAGES, NLSPATH, and PATH.

ENVIRONMENT VARIABLES

EXIT STATUS

The following exit values are returned:

0 All invocations of utility returned exit status 0.

1-125 A command line meeting the specified requirements could not be assembled, one or more of the invocations of utility returned a non-zero exit status, or some other error occurred.

126 The utility specified by utility was found but could not be invoked.

127 The utility specified by utility could not be found.

If a command line meeting the specified requirements cannot be assembled, the utility cannot be invoked, an invocation of the utility is terminated by a signal, or an invocation of the utility exits with exit status 255, the xargs utility will write a diagnostic message and exit without processing any remaining input.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>
xargs(1)

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSI</td>
<td>enabled</td>
</tr>
</tbody>
</table>

SEE ALSO

`echo(1), shell_builtins(1), attributes(5), environ(5)`
NAME
xgettext – extract gettext call strings from C programs

SYNOPSIS
xgettext [-ns] [-a [-x exclude-file]] [-c comment-tag] [-d default-domain]
       [-j] [-m prefix] [-M suffix] [-p pathname] - | filename...

xgettext -h

DESCRIPTION
The xgettext utility is used to automate the creation of portable message files (.po). A .po file contains copies of “C” strings that are found in ANSI C source code in filename or the standard input if ‘-’ is specified on the command line. The .po file can be used as input to the msgfmt(1) utility, which produces a binary form of the message file that can be used by application during run-time.

xgettext writes msgid strings from gettext(3C) calls in filename to the default output file messages.po. The default output file name can be changed by -d option. msgid strings in dgettext() calls are written to the output file domainname.po where domainname is the first parameter to the dgettext() call.

By default, xgettext creates a .po file in the current working directory, and each entry is in the same order that the strings are extracted from filenames. When the -p option is specified, the .po file is created in the pathname directory. An existing .po file is overwritten.

Duplicate msgids are written to the .po file as comment lines. When the -s option is specified, the .po is sorted by the msgid string, and all duplicated msgids are removed. All msgstr directives in the .po file are empty unless the -m option is used.

OPTIONS
The following options are supported:

- Add comment lines to the output file indicating file name and line number in the source file where each extracted string is encountered. These lines appear before each msgid in the following format:

```
# # File: filename, line: line-number
```

-s
Generate output sorted by msgids with all duplicate msgids removed.

-a
Extract all strings, not just those found in gettext(3C), and dgettext() calls. Only one .po file is created.

-c comment-tag
The comment block beginning with comment-tag as the first token of the comment block is added to the output .po file as # delimited comments. For multiple domains, xgettext directs comments and messages to the prevailing text domain.

-d default-domain
Rename default output file from messages.po to default-domain .po.
**xgettext(1)**

`-j` Join messages with existing message files. If a `.po` file does not exist, it is created. If a `.po` file does exist, new messages are appended. Any duplicate msgids are commented out in the resulting `.po` file. Domain directives in the existing `.po` file are ignored. Results not guaranteed if the existing message file has been edited.

`-m prefix` Fill in the msgstr with prefix. This is useful for debugging purposes. To make msgstr identical to msgid, use an empty string (""") for prefix.

`-M suffix` Fill in the msgstr with suffix. This is useful for debugging purposes.

`-p pathname` Specify the directory where the output files will be placed. This option overrides the current working directory.

`-x exclude-file` Specify a `.po` file that contains a list of msgids that are not to be extracted from the input files. The format of `exclude-file` is identical to the `.po` file. However, only the msgid directive line in `exclude-file` is used. All other lines are simply ignored. The `-x` option can only be used with the `-a` option.

`-h` Print a help message on the standard output.

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWloc</td>
</tr>
</tbody>
</table>

**SEE ALSO**

msgfmt(1), gettext(3C), attributes(5)

**NOTES**

xgettext is not able to extract cast strings, for example ANSI C casts of literal strings to `(const char *)`. This is unnecessary anyway, since the prototypes in `<libintl.h>` already specify this type.

In messages and translation notes, lines greater than 2048 characters are truncated to 2048 characters and a warning message is printed to stderr.
xstr – extract strings from C programs to implement shared strings

SYNOPSIS

xstr -c filename [-v] [-l array]
xstr [-l array]
xstr filename [-v] [-l array]

DESCRIPTION

xstr maintains a file called strings into which strings in component parts of a large program are hashed. These strings are replaced with references to this common area. This serves to implement shared constant strings, which are most useful if they are also read-only.

The command:

example% xstr -c filename

extracts the strings from the C source in name, replacing string references by expressions of the form &xstr[number] for some number. An appropriate declaration of xstr is prepended to the file. The resulting C text is placed in the file x.c, to then be compiled. The strings from this file are placed in the strings data base if they are not there already. Repeated strings and strings which are suffixes of existing strings do not cause changes to the data base.

After all components of a large program have been compiled, a file declaring the common xstr space called xs.c can be created by a command of the form:

example% xstr

This xs.c file should then be compiled and loaded with the rest of the program. If possible, the array can be made read-only (shared) saving space and swap overhead.

xstr can also be used on a single file. A command:

example% xstr filename

creates files x.c and xs.c as before, without using or affecting any strings file in the same directory.

It may be useful to run xstr after the C preprocessor if any macro definitions yield strings or if there is conditional code which contains strings which may not, in fact, be needed. xstr reads from the standard input when the argument ‘−’ is given. An appropriate command sequence for running xstr after the C preprocessor is:

example% cc -E name.c | xstr -c -
explema% cc -c x.c
example% mv x.o name.o

xstr does not touch the file strings unless new items are added; thus make(1S) can avoid remaking xs.o unless truly necessary.
xstr(1)

OPTIONS
- c filename  Take C source text from filename.
- v  Verbose: display a progress report indicating where
     new or duplicate strings were found.
- l array  Specify the named array in program references to
          abstracted strings. The default array name is xstr.

FILES
strings  data base of strings
x.c    massaged C source
xs.c    C source for definition of array “xstr*(rq
/tmp/xs*    temp file when xstr filename doesn’t touch strings

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWcsu</td>
</tr>
</tbody>
</table>

SEE ALSO
make(1S), attributes(5)

BUGS
If a string is a suffix of another string in the data base, but the shorter string is seen
first by xstr both strings will be placed in the data base, when just placing the longer
one there would do.

NOTES
Be aware that xstr indiscriminately replaces all strings with expressions of the form
\&xstr[number] regardless of the way the original C code might have used the string.
For example, you will encounter a problem with code that uses sizeof() to
determine the length of a literal string because xstr will replace the literal string with
a pointer that most likely will have a different size than the string’s. To circumvent this
problem:
  ▪ use strlen() instead of sizeof(); note that sizeof() returns the size of the
    array (including the null byte at the end), whereas strlen() doesn’t count the
    null byte. The equivalent of sizeof("xxx") really is (strlen("xxx"))+1.
  ▪ use #define for operands of sizeof() and use the define’d version. xstr
    ignores #define statements. Make sure you run xstr on filename before you run
    it on the preprocessor.

You will also encounter a problem when declaring an initialized character array of the
form
char x[] = "xxx";

xstr will replace xxx with an expression of the form \&xstr[number] which will not
compile. To circumvent this problem, use static char *x = "xxx" instead of
static char x[] = "xxx".
The `yacc` command converts a context-free grammar into a set of tables for a simple automaton that executes an LALR(1) parsing algorithm. The grammar may be ambiguous. Specified precedence rules are used to break ambiguities.

The output file, `y.tab.c`, must be compiled by the C compiler to produce a function `yyparse()`. This program must be loaded with the lexical analyzer program, `yylex()`, as well as `main()` and `yyerror()`, an error handling routine. These routines must be supplied by the user. The `lex(1)` command is useful for creating lexical analyzers usable by `yacc`.

The following options are supported:

- **-b file_prefix** Use `file_prefix` instead of `y` as the prefix for all output files. The code file `y.tab.c`, the header file `y.tab.h` (created when `-d` is specified), and the description file `y.output` (created when `-v` is specified), will be changed to `file_prefix.tab.c`, `file_prefix.tab.h`, and `file_prefix.output`, respectively.

- **-d** Generates the file `y.tab.h` with the `#define` statements that associate the `yacc` user-assigned “token codes” with the user-declared “token names”. This association allows source files other than `y.tab.c` to access the token codes.

- **-l** Specifying that the code produced in `y.tab.c` will not contain any `#line` constructs. This option should only be used after the grammar and the associated actions are fully debugged.

- **-P parser** Allows you to specify the parser of your choice instead of `/usr/ccs/bin/yaccpar`. For example, you can specify:

  ```
  example% yacc -P ~/myparser parser.y
  ```

- **-p sym_prefix** Uses `sym_prefix` instead of `yy` as the prefix for all external names produced by `yacc`. The names affected include the functions `yyparse()`, `yylex()` and `yyerror()`, and the variables `yylval`, `yychar` and `yydebug`. (In the remainder of this section, the six symbols cited are referenced using their default names only as a notational convenience.) Local names may also be affected by the `-p` option. However, the `-p` option does not affect `#define` symbols generated by `yacc`.

- **-Q[y|n]** The `-Qy` option puts the version stamping information in `y.tab.c`. This allows you to know what version of `yacc` built the file. The `-Qn` option (the default) writes no version information.
Compiles runtime debugging code by default. Runtime debugging code is always generated in `y.tab.c` under conditional compilation control. By default, this code is not included when `y.tab.c` is compiled. Whether or not the `-t` option is used, the runtime debugging code is under the control of `YYDEBUG`, a preprocessor symbol. If `YYDEBUG` has a non-zero value, then the debugging code is included. If its value is 0, then the code will not be included. The size and execution time of a program produced without the runtime debugging code will be smaller and slightly faster.

- `V`  Prints on the standard error output the version information for `yacc`.
- `-v` Prepares the file `y.output`, which contains a description of the parsing tables and a report on conflicts generated by ambiguities in the grammar.

**OPERANDS**

The following operand is required:

- `file`  A path name of a file containing instructions for which a parser is to be created.

**EXAMPLES**

**EXAMPLE 1** Accessing the yacc library

Access to the `yacc` library is obtained with library search operands to `cc`. To use the `yacc` library main:

```
example% cc y.tab.c -ly
```

Both the `lex` library and the `yacc` library contain `main`. To access the `yacc main`:

```
example% cc y.tab.c lex.yy.c -ly -ll
```

This ensures that the `yacc` library is searched first, so that its `main` is used.

The historical `yacc` libraries have contained two simple functions that are normally coded by the application programmer. These library functions are similar to the following code:

```c
#include <locale.h>
int main(void)
{
    extern int yyparse();

    setlocale(LC_ALL, "");

    /* If the following parser is one created by lex, the application must be careful to ensure that LC_CTYPE and LC_COLLATE are set to the POSIX locale. */
    (void) yyparse();
    return (0);
}
```

1690  man pages section 1: User Commands  •  Last Revised 20 Dec 1996
EXAMPLE 1 Accessing the yacc library  (Continued)

```c
#include <stdio.h>

int yyerror(const char *msg)
{
    (void) fprintf(stderr, "%s\n", msg);
    return (0);
}
```

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of yacc: LC_CTYPE, LC_MESSAGES, and NLSPATH.

yacc can handle characters from EUC primary and supplementary codesets as one-token symbols. EUC codes may only be single character quoted terminal symbols. yacc expects yylex() to return a wide character (wchar_t) value for these one-token symbols.

EXIT STATUS

The following exit values are returned:

- 0  Successful completion.
- >0  An error occurred.

FILES

- y.output  state transitions of the generated parser
- y.tab.c  source code of the generated parser
- y.tab.h  header file for the generated parser
- yacc.acts  temporary file
- yacc.debug  temporary file
- yacc.tmp  temporary file
- yaccpar  parser prototype for C programs

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWbtool</td>
</tr>
</tbody>
</table>

SEE ALSO

cc(1B), lex(1), attributes(5), environ(5)

DIAGNOSTICS

The number of reduce-reduce and shift-reduce conflicts is reported on the standard error output. A more detailed report is found in the y.output file. Similarly, if some rules are not reachable from the start symbol, this instance is also reported.
Because file names are fixed, at most one yacc process can be active in a given directory at a given time.

Users are encouraged to avoid using '$' as part of any identifier name.
NAME  yes – generate repetitive affirmative output

SYNOPSIS  yes  [expletive...]

DESCRIPTION  The yes utility repeatedly outputs y, or if expletive is given, it is output repeatedly,
followed by a newline. Multiple arguments are output separated by spaces and
followed by a newline. To terminate, type an interrupt character.

yes can be used to respond programatically to programs that require an interactive
response.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWesu</td>
</tr>
</tbody>
</table>

SEE ALSO  attributes(5)
ypcat(1)

NAME
ypcat – print values in a NIS database

SYNOPSIS
ypcat [-kx] [-d ypdomain] mname

DESCRIPTION
The ypcat command prints out values in the NIS name service map specified by
mname, which may be either a map name or a map nickname. Since ypcat uses the
NIS network services, no NIS server is specified.

Refer to ypfiles(4) for an overview of the NIS name service.

OPTIONS
-k Display the keys for those maps in which the values are null or the
    key is not part of the value. None of the maps derived from files
    that have an ASCII version in /etc fall into this class.
-d ypdomain Specify a domain other than the default domain.
-x Display map nicknames.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWnisu</td>
</tr>
</tbody>
</table>

SEE ALSO
ypmatch(1), ypfiles(4), attributes(5)
ypmatch – print the value of one or more keys from a NIS map

SYNOPSIS

ypmatch [-k] [-t] [-d domain] key [key...] mname

ypmatch -x

DESCRIPTION

ypmatch prints the values associated with one or more keys from the NIS’s name services map specified by mname, which may be either a map name or a map nickname.

Multiple keys can be specified; all keys will be searched for in the same map. The keys must be the same case and length. No pattern matching is available. If a key is not matched, a diagnostic message is produced.

OPTIONS

The following options are supported:

- `-k` Before printing the value of a key, print the key itself, followed by a colon (`:`).
- `-t` Inhibit map nickname translation.
- `-d domain` Specify a domain other than the default domain.
- `-x` Display the map nickname table. This lists the nicknames the command knows of, and indicates the map name associated with each nickname.

OPERANDS

The following operand is supported:

`mname` The NIS’s name services map

EXIT STATUS

The following exit values are returned:

0 Successful operation.
1 An error occurred.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWnisu</td>
</tr>
</tbody>
</table>

SEE ALSO

ypcat(1), ypfiles(4), attributes(5)

NOTES

ypmatch will fail with an RPC error message on yp operation if enough file descriptors are not available. The number of file descriptors should be increased if this occurs.
yp passwd – change your network password in the NIS database

yp passwd [username]

The yp passwd utility changes the network password associated with the user username in the Network Information Service (NIS) database. If the user has done a keylogin(1), and a publickey/secretkey pair exists for the user in the NIS publickey.byname map, yp passwd also re-encrypts the secretkey with the new password. The NIS password may be different from the local one on your own machine.

yp passwd prompts for the old NIS password, and then for the new one. You must type in the old password correctly for the change to take effect. The new password must be typed twice, to forestall mistakes.

New passwords must be at least four characters long, if they use a sufficiently rich alphabet, and at least six characters long if monocase. These rules are relaxed if you are insistent enough. Only the owner of the name or the super-user may change a password; superuser on the root master will not be prompted for the old password, and does not need to follow password construction requirements.

The NIS password daemon, rpc.yp passwd must be running on your NIS server in order for the new password to take effect.

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWnisu</td>
</tr>
</tbody>
</table>

SEE ALSO keylogin(1), login(1), nis+(1), nis passwd(1), passwd(1), getpwnam(3C), getsnarm(3C), secure_rpc(3NSL), nsswitch.conf(4), attributes(5)

WARNINGS Even after the user has successfully changed his or her password using this command, the subsequent login(1) using the new password will be successful only if the user’s password and shadow information is obtained from NIS. See getpwnam(3C), getsnarm(3C), and nsswitch.conf(4).

NOTES The use of yp passwd is discouraged, as it is now only a wrapper around the passwd(1) command, which should be used instead. Using passwd(1) with the -r nis option (see nis+(1)) will achieve the same results, and will be consistent across all the different name services available.

BUGS The update protocol passes all the information to the server in one RPC call, without ever looking at it. Thus, if you type your old password incorrectly, you will not be notified until after you have entered your new password.
ypwhich(1)

NAME
    ypwhich – return name of NIS server or map master

SYNOPSIS
    ypwhich [-d domain] [ [-t] -m [mname] | [-Vn]hostname]
    ypwhich -x

DESCRIPTION
    ypwhich returns the name of the NIS server that supplies the NIS name services to a
    NIS client, or which is the master for a map. If invoked without arguments, it gives the
    NIS server for the local machine. If hostname is specified, that machine is queried to
    find out which NIS master it is using.

    Refer to ypfiles(4) for an overview of the NIS name services.

OPTIONS
    -d domain        Use domain instead of the default domain.
    -t
        This option inhibits map nickname translation.
    -m mname         Find the master NIS server for a map. No hostname can be
                     specified with -m. mname can be a mapname, or a nickname for a
                     map. When mname is omitted, produce a list of available maps.
    -x
        Display the map nickname translation table.
    −Vn
        Version of ypbind, V3 is default.

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

    | ATTRIBUTE TYPE | ATTRIBUTE VALUE |
    |----------------|-----------------|
    | Availability   | SUNWnisu        |

SEE ALSO
    ypfiles(4), attributes(5)
Index

A
a new version of the network information name service — nis+, 982
a new version of the network information name service — NIS+, 982
a new version of the network information name service — nis, 982
accounting
search and print files — acctcom, 28
acctcom — search and print process accounting files, 28
adb — debugger, 31
addbib — create or extend bibliography, 32
adds /dev entries to give SunOS 4.x compatible names to SunOS 5.x devices —
ucblinks, 1594
alias — shell built-in functions to create your own pseudonym or shorthand for a command or series of commands, 34
aliases collected by sendmail — praliases, 1168
allocate — allocate devices, 37
answerbook2 — online documentation system, 40
appcert — examine application-level products for unstable use of Solaris interfaces, 41
apply changes to files — patch, 1085
apptrace — trace application function calls to Solaris shared libraries, 48
apropos — locate commands by keyword, 53
ar — maintain portable archive or library, 55
arch — display architecture of current host, 59
archive maintain a portable one across all machines — ar, 55
archives create tape archives, and add or extract files — tar, 1493
as — assembler, 60
asa — convert FORTRAN carriage-control output to printable form, 64
assembler — as, 60
at — execute commands at a later time, 66, 247
atq — display the jobs queued to run at specified times, 72
atrm — remove jobs spooled by at or batch, 73
audio file formats convert — audioconvert, 74
audio files play — audioplay, 78
record — audiorecord, 80
audioconvert — convert audio file formats, 74
audioplay — play audio files, 78
audiorecord — record an audio file, 80
authentication agent — ssh-agent, 1441
authentication key generation — ssh-keygen, 1445
auths — print authorizations granted to a user, 83
awk — pattern scanning and processing language, 85
B

banner — make posters, 90
basename — display portions of pathnames, 93
basename — strips affixes from path names, 91
batch — execute commands at a later time, 66, 247
bc — arbitrary precision arithmetic language, 94
bdiff — display line-by-line differences between pairs of large text files, 98
bfs — big file scanner, 99
bg — shell built-in functions to control process execution, 552
bibliography
create an inverted index to a bibliographic database — indexbib, 541
create or extend — addbib, 32
expand and insert references from a bibliographic database — refer, 1242
find references in a bibliographic database — lookbib, 747
format and print a bibliographic database — roffbib, 1268
sort a bibliographic database — sortbib, 1416
biff — mail notifier, 103
big file scanner — bfs, 99
binary file transmission
decode binary file — uudecode, 1616
encode binary file — uuencode, 1616
binary files
find printable strings — strings, 1453
locate — whereis, 1668
block count
for a file — sum, 1472
blocks, count a in file — sum, 1473
Bourne shell
— sh, 1375
Bourne shell commands
login command, 1385
Bourne shell variables, 1378
— CPATH, 1378
— HOME, 1378
— IFS, 1379
— MAIL, 1378
— MAILCHECK, 1378

Bourne shell variables (continued)
— MAILPATH, 1378
— PATH, 1378
— PS1, 1379
— PS2, 1379
— SHACCT, 1379
— SHELL, 1379
break — shell built-in functions to escape from or advance within a controlling while, for, foreach, or until loop, 104
build programs — make, 818

C

C compiler, 113
C language
— C preprocessor — cpp, 196
C language program
resolve and remove ifdef-ed lines from C program source — unifdef, 1602
C program verifier — lint, 703
C programming language
create C error messages — mkstr, 912
extract strings from C code — xstr, 1687
formats program in nice style using troff — vgrind, 1637
C shell
aliases — csh, 226
built-in commands — csh, 232
command and filename substitution — csh, 228
command execution — csh, 231
command line parsing — csh, 223
command substitution — csh, 228
control flow — csh, 231
environment variables and shell variables — csh, 241
event designators — csh, 224
expressions and operators — csh, 229
filename completion — csh, 222
filename substitution — csh, 229
history substitution — csh, 223
I/O redirection — csh, 226
initialization and termination — csh, 221
interactive operation — csh, 221
job control — csh, 232
lexical structure — csh, 223
C shell commands (continued)
 — repeat, 237
 — set, 237
 — setenv, 237
 — shift, 238
 — source, 238
 — stop, 238
 — suspend, 239
 — switch, 239
 — time, 239
 — umask, 239
 — unalias, 239
 — unhash, 239
 — unlimit, 240
 — unset, 240
 — unsetenv, 240
 — wait, 240
 — while, 240
 cal — display a calendar, 106
 calculator, desk
 — dc, 268
 calendar — reminder service, 107
 calendar
 display — cal, 106
 call-graph, display profile data — gprof, 508
 call rmmount to mount or unmount media —
 volrmmount, 1656
 cancel — cancel print requests, 109
 cancel user’s request for removable media that
 is not currently in drive — volcancel, 1652
 cat — concatenate and display files, 111
 cc — C compiler, 113
 cd — shell built-in functions to change the
 current working directory, 115
 CD read and write — cdrw, 118
 CDPATH variable — sh, 1378
 cdrw — CD read and write, 118
 change a user’s Kerberos password —
 kpasswd, 580
 change file access and modification times —
 touch, 1591
 change file access and modification times
 — settime, 1555
 — touch, 1555
 character translation — tr, 1564, 1569
 chdir — shell built-in functions to change the
 current working directory, 115
 check spelling — spell, 1420

Index 1701
check for media in a drive — volcheck, 1653
check path names — pathchk, 1090
checkeq — check eqn constructs, 341
checknr — check nroff/troff files, 124
chgrp — change the group ownership of a file, 125
chmod — change the permissions mode of a file, 129
chown — change owner of file, 135, 137
cksum — write file checksums and sizes, 159
clear — clear terminal screen, 167
cmp — compare two files, 168
ccocheck — (FMLI utility) communicate with a process, 180
cocreate — (FMLI utility) communicate with a process, 180
code formatter formats program in nice style using troff — vgrind, 1637
code set conversion utility — icnv, 538
codecreate — (FMLI utility) communicate with a process, 180
col — filters reverse line-feeds from two-column nroff text, 170
comm — select or reject lines common to two files, 172
command — execute a simple command, 174
cmd_options
parse — getopt, 492
parse — getoptcvt, 494
commands
display the last commands executed, in reverse order — lastcomm, 634
locate a command; display its pathname or alias — which, 1670
locate by keyword — apropos, 53
communications
connect to remote system — cu, 255
connect to remote system — tip, 1539
decode binary files — uudecode, 1616
encode binary files — uuencode, 1616
system to system command execution — uux, 1626
talk to another user — talk, 1490
UNIX-to-UNIX copy — uucp, 1612
communications (continued)
user interface to a remote system using the TELNET protocol — telnet, 1508
UUCP list of names — uuname, 1612
UUCP log — uulog, 1612
write to another user — write, 1677
Compaq Smart-2 EISA/PCI and Smart-2SL PCI Array Controller ioctl utility — smart2cfg, 1403
compare two files — diff, 278
compilers
C compiler — cc, 113
C program verifier — lint, 703
regular expression compile — regcmp, 1244
RPC protocol compiler — rpcgen, 1272
compress — compress files, 177
concatenate files and display them — cat, 111
configure LLC2 interface parameters — llc2_config, 709
configure runtime linking environment — crle, 206
connect to remote system — cu, 255
construct argument lists and invoke utility — xargs, 1680
continue — shell built-in functions to escape from or advance within a controlling while, for, foreach, or until loop, 104
control
audio mixer control — mixerctl, 906
control line printer — lpc, 755
control tracing and manipulate probe points in a process or the kernel — prex, 1174
convert binary log file to Common Log File format — ncab2clf, 964
convert binary TNF file to ASCII — tnfdump, 1548
convert FORTRAN carriage-control output to printable form — asa, 64
convert Red Hat Package (RPM) to cpio archive — rpm2cpio, 1277
convert units — units, 1606
coproc — (FMLI utility) communicate with a process, 180
copy
archives — cpio, 188
files — cp, 184
core image
  of running processes — gcore, 453
coreceive — (FMLI utility) communicate with a process, 180
cosend — (FMLI utility) communicate with a process, 180
count blocks in file — sum, 1473
count lines, words, characters in file — wc, 1664
cp — copy files, 184
cpio — copy archives, 188
cpp — C preprocessor, 196
cputrack — monitor process and LWP behavior using CPU performance counters, 202
create
  bibliography — addbib, 32
  create a tags file for use with ex and vi — ctags, 252
create new task or change task or project of running process — newtask, 978
crle — configure runtime linking environment, 206
crontab — user crontab file, 216
crypt — encrypt, 220
csh — shell command interpreter with a C-like syntax, 221
csplit — split files based on context, 247
c — spawn login to a remote terminal, 250
ctags — create a tags file for use with ex and vi, 252
cu — connect to remote system, 255
curve, smooth
  interpolate — spline, 1423
cut — cut out selected fields of each line of a file, 261
del — delete a file, 239
deallocate — deallocate devices, 272
debug tools
  debugger — adb, 31
develop binary file — uudecode, 1616
develop files
  — crypt, 220
decrypt — crypt, 220
define locale environment — localedef, 729
dependencies, dynamic
  of executable files or shared objects — ldd, 672
deroff — remove nroff, troff, tbl and eqn constructs, 274
describe command — whatis, 1667
describe instruction set architectures — isainfo, 549
desk calculator
  — dc, 268
determine which variant instruction set is optimal to use — optisa, 1069
devices
  allocation — allocate, 37
deallocation — deallocate, 272
  eject media device from drive — eject, 331
  list_devices — list_devices, 705
df — display status of disk space on file systems, 275
dhcpinfo — display value of parameters received through DHCP, 276
dictionary, system
  find words — look, 746
diff — compare two files, 278
diff
  3-way — dif3, 282
  big — bdiff, 98
diff command
  side-by-side — sdiff, 1344
dif3 — display line-by-line differences between three text files, 282
diffmk — mark differences between versions of a troff input file, 284
digestp — frontends to the mp Text to PDL (Printer Description Language) pretty print filter, 791
dircmp — compares contents of directories, 285
directories
  compare contents — dircmp, 285
  list contents — ls, 772
directories (continued)
list contents of — ls, 778
make — mkdir, 908
make link to — ln, 722
print working directory name — pwd, 1228
remove — rmdir, 1256
dirname — delivers all but last level of path name, 91
dirs — shell built-in functions to change the current working directory, 115
dis — object code disassembler, 286
disable — disable LP printers, 337
disassembler
object code — dis, 286
disk usage
summary — du, 297
display editor — vi, 1641
display
a list of all valid user names — dispuid, 289
architecture of current host — arch, 59
call-graph profile data — gprof, 308
contents of directory — ls, 772
current news — news, 977
— date, 264
disk usage — du, 300
dynamic dependencies of executable files or shared objects — ldd, 672
effective user name — whoami, 1674
file names — ls, 778
first few lines of files — head, 525
group membership of user — groups, 520
how long the system has been up — uptime, 1610
identifier of current host — hostid, 536
last commands executed, in reverse order — lastcomm, 634
list of all valid group names — dispgid, 288
login and logout information about users and terminals — last, 632
name of current host — hostname, 537
name of the user running the process — logname, 744
printer queue — lpq, 759
process status — ps, 1222
processor type of current host — mach, 786
selected lines from file — sed, 1353
size or sizes of a page of memory — pagesize, 1073

display (continued)
status of disk space on file system — df, 275
status of local hosts — ruptime, 1288
status of network hosts — rup, 1286
users on system — users, 1611
working directory name — pwd, 1228
display discretionary file information —
getfacl, 486
display information about currently logged-in users — w, 1659
display information about the address space of a process — pmap, 1136
display names and references bound in FNS context — fnlist, 426
display package parameter values —
pkgparam, 1126
display profile data — prof, 1206
display reference bound to FNS name — fnlookup, 428
display the internal versioning information of dynamic objects — pvs, 1225
display the native instruction sets executable on this platform — isalist, 551
display value of parameters received through
DHCP — dhcpcfg, 276
document production
check spelling — spell, 1420
check nroff/troff files — checknr, 124
create an inverted index to a bibliographic database — indexbib, 541
create or extend bibliography — addbib, 32
eliminate .so’s from nroff input —
soelim, 1405
expand and insert references from a bibliographic database — refer, 1242
filters reverse line-feeds from two-column nroff text — col, 170
find references in a bibliographic database —
lookbib, 747
format and print a bibliographic database — roffbib, 1268
format documents for display or line-printer — nroff, 1058
format tables for nroff or troff — tbl, 1504
mark differences between versions of a troff input file — diffmk, 284
remove nroff, troff, tbl and eqn constructs —
deroff, 274
document production (continued)
simple text formatters — fmt, 415
sort a bibliographic database — sortbib, 1416
troff postprocessor for PostScript printers — dpost, 294
typeset mathematics — eqn, 341
typeset or format documents — troff, 1572
DOS
convert text file from DOS format to ISO format — dos2unix, 290
convert text file from ISO format to DOS format — unix2dos, 1608
dos2unix — convert text file from DOS format to ISO format, 290
download — host resident PostScript font downloder, 292
download — prepends host resident PostScript fonts to files, 292
dpost — troff postprocessor for PostScript printers, 294
draw graph — graph, 513
du — summarize disk usage, 297
du — display disk usage per directory or file, 300
dump — dump selected parts of an object file, 302
dump selected parts of an object file — dump, 302
dump selected parts of an object file — elfdump, 335
dumpcs — show codeset table for the current locale, 305
dumpkeys — dump keyboard translation tables, 725
echo — echo arguments, 306
echo — (FMLI utility) put string on virtual output, 311
echo — echo arguments to standard output, 310
ed — text editor, 312
edit — text editor, 324
editing text
sed — stream editor, 1353
egrep — search a file for a pattern using full regular expressions, 328
eject — eject media device from drive, 331
elfdump — dump selected parts of an object file, 335
enable — enable LP printers, 337
encode binary file — uuencode, 1616
encode files
— crypt, 220
encryption key, user
change — chkey, 127
eval — set environment for command invocation, 339
environment
display variables — printenv, 1186
set terminal characteristics — tset, 1583
environment variables, global FMLI, 1364
eqn — mathematical typesetting, 341
eqn
— remove nroff, troff, tbl and eqn constructs — deroff, 274
equations
typeset mathematics — eqn, 341
error — analyze error messages, 346
eval — shell built-in functions to execute other commands, 359
evaluate condition(s) — test, 1518
ex — text editor, 350
examine application-level products for unstable use of Solaris interfaces — appcert, 41
exec — shell built-in functions to execute other commands, 359
execute a command in a profile — pfsh, 1111
execute a command in a profile — pfexec, 1111
execute a command in a profile — pfksh, 1111
execute commands at a later time — at, 66, 247
execute commands at a later time — batch
— batch, 66, 247
execute a simple command — command, 174
execute command in extended attribute name space — runat, 1283
exit — shell built-in functions to enable the execution of the shell to advance beyond its sequence of steps, 361
expand — expand TAB characters to SPACE characters, 363
export — shell built-in functions to determine the characteristics for environmental variables of the current shell and its descendents, 1359
exportfs — translates exportfs options to share/unshare commands, 365
expr — evaluate arguments as an expression, 366
expr — evaluate expressions, 369
expression evaluation — expr, 369
exstr — extract strings from source files, 372
extract kernel probes output into a trace file — tnfxtract, 1553
extract strings from C code — xstr, 1687

F
face — executable for the Framed Access Command Environment Interface, 376
factor — obtain the prime factors of a number, 377
false — provide truth values, 1575
fastboot — reboot system without checking disks, 378
fasthalt — halt system without checking disks, 378
fc — shell built-in functions to re-use previous command-lines from the current shell, 527
fdformat — format floppy diskette or PCMCIA memory card, 379
fg — shell built-in functions to control process execution, 552
fgrep — search file for fixed-character string, 383
file — determine file type, 385
file — change ownership — chown, 137
determine type of — file, 387
display names — ls, 778
files perusal filter for CRTs — pg, 1112
make link to — ln, 722
print — lpr, 761
strip affixes — basename, 93
sum — sum and count blocks in file, 1473
update last modified date of — touch, 1558
file — get file type, 387
file system
display status of disk space — df, 275
make hard or symbolic links to files — ln, 719
where am I — pwd, 1228
file transfer program — ftp, 440
filep — frontends to the mp Text to PDL (Printer Description Language) pretty print filter, 791
files
change owner of file — chown, 135
change the permissions mode of a file — chmod, 129
compare two files — cmp, 168
compress — compress, 177
compress files — pack, 1070
compress files — pcat, 1070
concatenate and display — cat, 111
copy — cp, 184
copy archives — cpio, 188
crypt — encrypt/decrypt, 220
cut out selected fields of each line of a file — cut, 261
display uncompressed files but leaves compressed files intact — zcat, 177
display a count of lines, words and characters in a file — wc, 1664
display first few lines — head, 525
display last part — tail, 1487
display line-by-line differences between pairs of large text files — bdiff, 98
display line-by-line differences between three text files — diff3, 282
expand compressed files — unpack, 1070
extract SCCS version information from a file — what, 1666
— find, 396
mark differences between versions of a troff input file — diffmk, 284
merge same lines of several files or subsequent lines of one file — paste, 1082
move — mv, 938
print checksum and block count for a file — sum, 1472
print differences between two files side-by-side — sdiff, 1344
files (continued)
remove — rm, 1256
search a file for a pattern — grep, 515
search file for fixed-character string — fgrep, 383
search for a pattern using full regular
expressions — egrep, 328
sort or merge — sort, 1409
split a file into pieces — split, 1424
strip affixes from path names — basename, 91
transfer to and from a remote machine — tftp, 1530
uncompress — uncompress, 177
filesync — synchronize files and
directories, 389
Multiple Nomadic Machines, 391
Rules File, 390
filofaxp — frontends to the mp Text to PDL
(Printer Description Language) pretty print
filter, 791
find — find files, 396
find or signal processes by name and other
attributes — pgrep, 1117
— pkill, 1117
fmlcut — (FMLI utility) cut out columns from a
table or fields from each line of a file, 406
fmlexpr — (FMLI utility) evaluate arguments as an
expression, 408
fmlgrep — (FMLI utility) search afi le for a pattern, 410
FMLI (continued)
fmlgrep — search afi le for a pattern, 410
fml — invoke fmli, 412
getfrm — returns the current frameID
number, 490
getitems — returns a list of currently marked
menu items, 491
indicator — displays application specific
alarms or working indicator, or both, on
FMLI banner line, 540
message — puts arguments on FMLI
message line, 904
pathconv — converts an alias to its
pathname, 1093
readfile, longline — reads file, gets longest
line, 1240
regex — match patterns against a
string, 1246
reinit — changes the descriptors in the
initialization file, 1248
reset — (FLMI utility) changes the entry in a
field of a form to its default value, 1252
run — runs a program, 1281
set, unset — set and unset local or global
environment variables, 1364
setcolor — redefine or create a color, 1366
shell — run a command using shell, 1393
test — evaluates the expression
expression, 1528
vsg — synchronize a co-process with the
controlling FMLI application, 1658
fmt — simple text formatters, 415
fnattr — update and examine attributes
associated with FNS named object, 421
fnlist — display names and references bound in
FMLI context, 426
fnlookup — display reference bound to FNS
name, 428
fnrename — rename the binding of an FNS
name, 429
FNS
display names and references — fnlist, 426
display reference bound to FNS name —
fnlookup, 428
search for FNS objects — fnsearch, 430
update attributes — fnattr, 421
fnsearch — search for FNS objects with
specified attributes, 430
FNS (continued)
Displaying Selected Attributes, 432
Extended Operations, 434
Filter Arguments, 432
Grammar of Filter Expressions, 434
Logical Operators, 431
Relational Operators, 431
Simple Filter Expressions, 431
Wildcarded Strings, 433
fnunbind — unbind the reference from an FNS name, 436
fold — fold long lines, 437
fonts
  preprends host resident PostScript fonts to files — download, 292
force a defunct process to be reaped by its parent — preap, 1172
format floppy diskette or PCMCIA memory card — fdformat, 379
formatters, text
  — fmt, 415
Forms and Menu Language Interpreter, See FMLI
FORTRAN
  create a tags file for use with ex and vi — ctags, 252
Framed Access Command Environment, See face
frameID number (FMLI utility) — getfrm, 490
franklinp — frontends to the mp Text to PDL (Printer Description Language) pretty print filter — digestp, 791
from — sender of mail messages, 439
frontends to the mp Text to PDL (Printer Description Language) pretty print filter — mailp, 791
frontends to the mp Text to PDL (Printer Description Language) pretty print filter — newsp, 791
frontends to the mp Text to PDL (Printer Description Language) pretty print filter — timemanp, 791
frontends to the mp Text to PDL (Printer Description Language) pretty print filter — timesysp, 791
ftp — file transfer program, 440
ftpcount — show current number of users in each FTP Server class, 451
ftpwho — show current process information for each FTP Server user, 452
function calls
  trace application function calls to Solaris shared libraries — aptrace, 48
G
gcore — get core images of running processes, 453
genCat — generate a formatted message catalog, 454
generate iconv code conversion tables — geniconvtbl, 457
generate LLC2 configuration files — llc2_autoconfig, 708
generate message source file from source files — genmsg, 475
generate programs for lexical tasks — lex, 686
generate repetitive affirmative output — yes, 1693
geniconvtbl — generate iconv code conversion tables, 457
genmsg — generate message source file from source files, 475
  Auto Message Numbering, 475
  Comment Extraction, 475
  Invocation, 475
  Testing, 475
get configuration values — getconf, 481
get locale-specific information — locale, 726
get or set the resource controls of running processes, tasks, and projects — prctl, 1169
get or set the resource limits of running processes
genmsg — generate message source file from source files (continued)
    — plimit, 1132
getconf — get configuration values, 481
getfacl — display discretionary file information, 486
getfrm — (FMLI utility) returns the current frameID number, 490
getitems — (FMLI utility) returns a list of currently marked menu items, 491
getopt — parse command options, 492
getoptcvt — parse command options, 494
getopts — shell built-in function to parse command-line options, 497
gettext — retrieve text string from message database, 503
gettxt — retrieve text string from message database, 505
glob — shell built-in function to expand a word list, 507
goto — shell built-in functions to enable the execution of the shell to advance beyond its sequence of steps, 361
gprof — call-graph profile, 508
graph — draw graph, 513
graphics filters for plotters — plot, 1134
graphics filters for plotters — tplot, 1559
graphics interpolate smooth curve — spline, 1423
grep
    search a file for a pattern — grep, 515
    search a file for a pattern using full regular expressions — egrep, 328
    search file for fixed-character string — fgrep, 383
group IDs
    change real and effective — newgrp, 975
    change the group ownership of a file — chgrp, 125
    display a list of all valid group names — dispgid, 288
    prompts for group ID — ckgid, 141
    provides error message for group ID — errgid, 141
    provides help message for group ID — helpgid, 141
    validates group ID — valgid, 141
    groups — display group membership, 521
groups — print group membership of user, 520
grpck — check group database entries, 522

H
halt system without checking disks — fasthalt, 378
hash — shell built-in functions to evaluate the internal hash table of the contents of directories, 523
hashstat — shell built-in functions to evaluate the internal hash table of the contents of directories, 523
head — display first few lines of files, 525
history — shell built-in functions to re-use previous command-lines from the current shell, 527
HOME variable — sh, 1378
host machines, local
    show status — ruptime, 1288
    who is logged in — rwho, 1292
host machines, remote
    display status of network hosts (RPC version) — rup, 1286
    who is logged in — rusers, 1291
host resident PostScript font downloader — download, 292
hostid — display host ID, 536
hostname — display host name, 537

I
i386 — get processor type truth value, 787
iAPX286 — get processor type truth value, 787
IFS variable — sh, 1379
indicator — (FMLI utility) displays application specific alarms or working indicator, or both, on FMLI banner line, 540
indxbib — create an inverted index to a bibliographic database, 541
install — install files, 542
instruction set, determining which variant is optimal to use — optisa, 1069
integer
    prompts for an integer — ckgid, 143
integer (continued)
  provides error message for integer —
    errint, 143
  provides help message for integer —
    helpint, 143
  validates an integer — valint, 143
integer, range
  prompts for an integer within a specified
    range — ckrange, 153
  provides error message for integer within a
    specified range — errange, 153
  provides help message for integer within a
    specified range — helprange, 153
  validate an integer within a specified range —
    valrange, 153
interactive message processing system —
  Mail, 789
interactive message processing system —
  Mail, 789
Internet
  transfer files to and from a remote machine
    — tftp, 1530
  transfer of files to and from remote network
    sites — ftp, 440
  user name directory service — whois, 1676
interprocess communication
  remove a message queue, semaphore set, or
    shared memory ID — ipcrm, 544
  report status — ipcs, 545
invoke a command with an altered scheduling
  priority — nice, 980
ipcrm — remove a message queue, semaphore
  set, or shared memory ID, 544
ipcs — report inter-process communication
  facilities status, 545
isainfo — describe instruction set
  architectures, 549
isalist — display the native instruction sets
  executable on this platform, 551

K
  kbd — manipulate the state of keyboard or
    display the type of keyboard or change the
    default keyboard abort sequence effect, 562
Kerberos keytab maintenance utility —
  ktutil, 630
Kerberos login utility
    — kinit, 573
Kerberos tickets
  destroy — kdestroy, 565
  list currently held — klist, 578
keyboard
  load and dump keyboard translation tables
    — loadkeys, dumpkeys, 725
  manipulate the state of keyboard or display
    the type of keyboard or change the default
    keyboard abort sequence effect —
    kbd, 562
keylogin — decrypt and store secret key with
  keyserv, 566
keylogout — delete stored secret key with
  keyserv, 568
keywords
  prompts for and validates a keyword —
    ckkeywd, 148
kill — terminate a process by default, 569
Korn shell commands
  login command, 618
KornShell
  aliasing — ksh, 583
  arithmetic evaluation — ksh, 599
  blank interpretation — ksh, 598
  command execution — ksh, 606
  command re-entry — ksh, 607
  command substitution — ksh, 586
  commands — ksh, 581
  comments — ksh, 583
  conditional expressions — ksh, 599
  definitions — ksh, 581
  emacs editing mode — ksh, 608
  environment — ksh, 603
  file name generation — ksh, 598
  functions — ksh, 603
  I/O — ksh, 601
  in-line editing options — ksh, 607
  invocation — ksh, 626
  jobs — ksh, 605
  jobs — shell builtins, 554

J
  jobs — shell built-in functions to control process
    execution, 552
  join — relational database operator, 559
  jsh — the job control shell command
    interpreter, 1375
KornShell (continued)
  parameter substitution — ksh, 589
  process substitution — ksh, 589
  prompting — ksh, 599
  quoting — ksh, 598
  restricted command and programming language — rksh, 581
  signals — ksh, 606
  special commands — ksh, 614
  tilde substitution — ksh, 585
  vi editing mode — ksh, 611
kpasswd — change a user’s Kerberos password, 580
ksh — KornShell, a standard command and programming language, 581
ktutil — Kerberos keytab maintenance utility, 630

L
languages
  C compiler — cc, 113
  C preprocessor — cpp, 196
  C program verifier — lint, 703
  create C error messages — mkstr, 912
  extract strings from C code — xstr, 1687
last — display login and logout information about users and terminals, 632
lastcomm — display the last commands executed, in reverse order, 634
ld — link-editor for object files, 636
ld — link editor, 648
ld.so.1 — runtime linker for dynamic objects, 677
ldap — LDAP as a naming repository, 649
LDAP as a naming repository — ldap, 649
ldap delete entry tool —ldapdelete, 653
ldap entry addition and modification tools
  — ldapadd, 660
  — ldapmodify, 660
ldap modify entry RDN tool —
  ldapmodrdn, 664
ldap search tool — ldapssearch, 667
ldapadd — ldap entry addition and modification tools, 660
ldapdelete — ldap delete entry tool, 653
ldapmodify — ldap entry addition and modification tools, 660
ldapmodrdn — ldap modify entry RDN tool, 664
  Input Format, 665
ldapssearch — ldaps search tool, 667
  Output Format, 667
ld — list dynamic dependencies of executable files or shared objects, 672
let — shell built-in function to evaluate one or more arithmetic expressions, 685
lex — generate programs for lexical tasks, 686
  Actions in lex, 693
  create a tags file for use with ex and vi —
    ctags, 252
lex — generate programs for lexical tasks
  Definitions in lex, 688
  Output Files, 687
  Regular Expressions in lex, 690
  Rules in lex, 689
  Stderr, 686
  Stdout, 686
  User Subroutines in lex, 690
library archive
  find ordering relation for an object or library archive — lorder, 748
limit — set or get limitations on the system resources available to the current shell and its descendents, 698
  csh, 698
  ksh, 698
  sh, 698
  sh/ksh, 700
  /usr/bin/ulimit, 698
line — read one line from standard input and write to standard output, 702
line numbering filter — nl, 1046
line printer control — lpc, 755
link
  make hard or symbolic links to files —
    ln, 719
link editor — ld, 648
link-editor — ld, 636
lint — C program verifier, 703
list
  contents of directory — ls, 772
  file names — ls, 778
logout
display login and logout information about users and terminals — last, 632
look — find words in the system dictionary or lines in a sorted list, 746
lookbib — find references in a bibliographic database, 747
lorder — find ordering relation for an object or library archive, 748
lp — send requests to a print service, 749
LP print services
cancel requests — cancel, 109
cancel line printer — lp, 755
display printer queue — lpq, 759
generate printer test pattern — lptest, 771
print files — lp, 749
print files (BSD) — lpr, 761
remove print jobs — lprm, 765
lpc — line printer control, 755
lpq — display printer queue, 759
lpr — print files, 761
lprm — remove print jobs, 765
lpstat — print information about the status of the print service, 767
lptest — generate printer test pattern, 771
ls — list contents of directory, 772
ls — list files, 778

M
m4 — macro processor, 781
mach — display processor type of current host, 786
machid — get processor type truth value, 787
machine IDs
generate processor type truth value — machid, 787
macro processor — m4, 781
magnetic tape
backspace files — mt, 935
backspace records — mt, 935
copy — tcpy, 1506
erase — mt, 935
forward space files — mt, 935
forward space records — mt, 935
generate unit status — mt, 935
manipulate — mt, 935
magnetic tape (continued)
  place unit off-line — mt, 935
  retension — mt, 935
  rewind — mt, 935
  skip backward files — mt, 935
  skip backward records — mt, 935
  skip forward files — mt, 935
  skip forward records — mt, 935
  write EOF mark on — mt, 935
mail — interactive message processing
  system, 789
Mail — interactive message processing
  system, 789
MAIL variable — sh, 1378
mail
  automatic replies — vacation, 1630
mail aliases
  aliases— system mail, 1168
mail services
  mail notifier — biff, 103
  sender of mail messages — from, 439
mail utilities
  statistics — mailstats, 795
mailbox
  storage for incoming mail — mailx, 797
MAILCHECK variable — sh, 1378
mailcompat — provide SunOS compatibility for
  Solaris mailbox format, 790
mailp — frontends to the mp Text to PDL
  (Printer Description Language) pretty print
  filter, 791
mailstats — mail delivery statistics, 795
mailx — interactive message processing
  system, 797, 817
mailx commands (continued)
  — discard, 802
  — dp, 802
  — dt, 802
  — echo, 802
  — edit, 802
  — else, 803
  — endif, 803
  — exit, 802
  — field, 802
  — file, 802
  — folder, 802
  — Followup, 803
  — followup, 803
  — from, 803
  — group, 801
  — headers, 803
  — help, 803
  — hold, 803, 805
  — if, 803
  — ignore, 802
  — inc, 803
  — list, 804
  — load, 804
  — mail, 804
  — mbox, 804
  — More, 804
  — New, 804
  — new, 804
  — next, 805
  — Page, 804
  — pipe, 805
  — preserve, 803, 805
  — Print, 805, 807
  — print, 805, 807
  — Put, 805
  — put, 805
  — quit, 805
  — Reply, 806
  — reply, 805
  — replyall, 806
  — repliesender, 805
  — Respond, 806
  — respond, 805
  — retain, 806
  — Save, 806
  — save, 806
  — set, 806
mailx commands (continued)
— shell, 807
— size, 807
— source, 807
— top, 807
— touch, 807
— Type, 805, 807
— type, 805, 807
— unalias, 807
— undelete, 807
— undiscard, 807
— ungroup, 807
— unignore, 807
— Unread, 804, 807
— unread, 804, 807
— unretain, 807
— unset, 807
— version, 808
— visual, 808
— write, 808
— xit, 808

maintain groups of programs —
sysV-make, 1476
make — maintain, update, and regenerate related programs and files
Appending to a Macro, 827
Bourne Shell Constructs, 847
Clearing Special Targets, 826
Command Dependencies, 826
Command Execution, 846
Command Substitutions, 847
Conditional Macro Definitions, 829
Dynamic Macros, 829
Global, 822
Hidden Dependencies, 826
Implicit Rules, 832
implicit rules, list of make/make.rules, 846
Library Maintenance, 846
Macros, 823, 826
Makefile Target Entries, 821
Pattern Matching Rules, 833
Pattern Replacement Macro References, 827
Predefined Macros, 829
Reading Makefiles and the Environment, 821
Rules, 824
Signals, 847
Special Characters, 822

make — maintain, update, and regenerate related programs and files (continued)
Special-Function Targets, 824
Special-Purpose Macros, 827
Suffix Replacement Macro References, 827
Suffix Rules, 833
make
System V version of make —
sysV-make, 1476
make — maintain, update, and regenerate related programs and files
Targets and Dependencies, 822
The Suffixes List, 846
man — online display of reference pages, 853
manual pages
accessing — man, 853
describe command — whatis, 1667
locate — whereis, 1668
matrix display program for PostScript printers — postmd, 1150
mbox
storage file for read mail — mailx, 797
mconnect — open connection to remote mail server, 859
mcs — manipulate the comment section of an object file, 860
mdb — modular debugger, 862
menu item
builds a menu and prompts user to choose one item from menu — ckitem, 145
provides error message for menu item — erritem, 145
provides help message for menu item — helpitem, 145
menu items, FMLI
returns a list of — getitems, 491
mesg — permit or deny messages via write, 903
message — puts arguments on FMLI message line, 904
messages
create message object file — msgfmt, 929
display contents of, or search for a text string in, message data bases — srctxtx, 1426
display on stderr or system console — fmtmsg, 416
extract gettext call strings — xgettext, 1685
messages (continued)
generate a formatted message catalog —
gencat, 454
permit or deny messages via write —
mesg, 903
retrieve text string from message database —
gettext, 503
mixerctl — audio mixer control, 906
mkdir — make directories, 908
mkmsgs — create message files for use by
gettxt, 910
mkstr — create C error messages, 912
modify the Access Control List (ACL) for a file
or files — setfacl, 1367
modular debugger — mdb, 862
monitor process and LWP behavior using CPU
performance counters — cputrack, 202
more — browse through a text file, 914
mp — text to PDL (Printer Description
Language) pretty print filter, 920
mpss.so.1 — shared object for setting preferred
page size, 926
msgfmt — create message object file, 929
mt — manipulate magnetic tape, 935
mv — move files, 938

N

awk — pattern scanning and processing
language, 941
Actions, 941
Arithmetic Functions, 941
Expression Patterns, 941
Expressions in awk, 941
Functions, 941
Input/Output and General Functions, 941
Output Statements, 941
Pattern Ranges, 941
Patterns, 941
Regular Expressions, 941
Special Patterns, 941
String Functions, 941
User-defined Functions, 941
/us/bin/nawk, 941
/us/sbin/awk, 941
/us/xpg4/bin/awk, 941
Variables and Special Variables, 941

NCA — the Network Cache and Accelerator
(NCA), 962
nca — the Network Cache and Accelerator
(NCA), 962
ncab2clf — convert binary log file to Common
Log File format, 964
ncakmod — start or stop the NCA kernel
module, 966
necn — mathematical typesetting, 341
netscape — start Netscape Communicator for
Solaris, 967
newform — change the format of a text
file, 972
newgrp — changes a user’s group ID, 975
newgrp — shell built-in function to allow new
group permissions to the user, 975
news — print news items, 977
newsp — frontends to the mp Text to PDL
(Printer Description Language) pretty print
filter, 791
newtask — create new task or change task or
project of running process, 978
NFS, secure
decrypt and store secret key with keyserv —
keylogin, 566
delete stored secret key with keyserv—
keylogout, 568
nice — invoke a command with an altered
scheduling priority, 980
nice
change process nice value — renice, 1249
nice — invoke a command with an altered
scheduling priority
csh Built-in, 980
nis — a new version of the network information
name service, 982
NIS
change login password in —
yppasswd, 1696
print the value of one or more keys from a
NIS map — ypmatch, 1695
print values in a NIS database —
ypcat, 1694
return name of NIS server or map master —
ywhich, 1697
nis+ — a new version of the network
information name service, 982
NIS+ — a new version of the network information name service, 982

NIS+ Authentication — nis+, 989
Authorization — nis+, 989
change access rights on a NIS+ object — nischmod, 1002
change password information — nispasswd, 1030
change the group owner of a NIS+ object — nischgrp, 1000
change the owner of a NIS+ object — nischown, 1005
change the time to live of a NIS+ object — nischttl, 1007
Concatenation Path — nis+, 985
create NIS+ directories — nismkdir, 1024
Directories and Domains — nis+, 988
Directory Authorization — nis+, 990
display NIS+ defaults — nisdefaults, 1009
display NIS+ error messages — niserror, 1012
display tables — niscat, 997
Grammar — nis+, 984
Group Names — nis+, 988
group administration — nisgrpadm, 1013
Indexed Names — nis+, 983
list the contents of a NIS+ directory — nisls, 1019
Name Expansion — nis+, 984
Namespaces — nis+, 986
NIS+ Administrative Commands — nis+, 993
NIS+ Directory Object — nis+, 982
NIS+ Files and Directories — nis+, 995
NIS+ Group Object — nis+, 982
NIS+ Link Object — nis+, 983
NIS+ Programming API — nis+, 994
NIS+ Table Object — nis+, 982
NIS+ User Commands — nis+, 992
Principal Names — nis+, 986
remove directories — nisrmdir, 1036
remove objects — nisrm, 1034
return the state of the NIS+ namespace using a conditional expression — nistest, 1044
Simple Names — nis+, 983
symbolically link NIS+ objects — nisln, 1017

NIS+ (continued)
Table Authorization — nis+, 991
table administration tool — nistbladm, 1038
utilities for searching NIS+ tables — nismatch, nisgrep, 1021
niscat — display NIS+ tables, 997
nischgrp — change the group owner of a NIS+ object, 1000
nischmod — change access rights on a NIS+ object, 1002
nischown — change the owner of a NIS+ object, 1005
nishttl — change the time to live of a NIS+ object, 1007
nisdefaults — display NIS+ defaults, 1009
niserror — display NIS+ error messages, 1012
nisgrep — utility for searching NIS+ tables, 1021
nisgrpadm — NIS+ group administration command, 1013
nisln — symbolically link NIS+ objects, 1017
nisls — list the contents of a NIS+ directory, 1019
nismatch — utility for searching NIS+ tables, 1021
nismkdir — create a NIS+ directory, 1024
nism — remove NIS+ objects, 1034
nismdir — remove a NIS+ directory, 1036
nistbladm — administer NIS+ tables, 1038
nistest — return the state of the NIS+ namespace using a conditional expression, 1044
nl — number lines, 1046
nm — print name list of an object file, 1049
nohup — run a command immune to hangups, 1054
notify — shell built-in functions to control process execution, 552
notify user that volume requested is not in the CD-ROM or floppy drive — volmissing, 1655
nroff — format documents for display or line-printer, 1058
nroff utilities — check nroff and troff files — checknr, 124
eliminate .so’s from nroff input — soelim, 1405
nroff utilities (continued)
  filters reverse line-feeds from two-column
  nroff text — col, 170
  format tables — tbl, 1504
  remove nroff, troff, tbl and eqn constructs —
  deroff, 274

O
  object archive
    find ordering relation for an object or library
    archive — lorder, 748
  object files
    find printable strings — strings, 1453
    manipulate the comment section —
    mcs, 860
    print section sizes in bytes — size, 1400
    strip symbol table, debugging and line
    number information — strip, 1455
  octal dump
    — od, 1061
    od — octal dump, 1061
  on — execute a command on a remote system,
    but with the local environment, 1067
  onintr — shell built-in functions to respond to
    (hardware) signals, 1570
  online documentation system
    — answerbook2, 40
  online reference pages — man, 853
  optisa — determine which variant instruction
    set is optimal to use, 1069

P
  pack — compress files, 1070
  page — page through a text file, 914
  pagesize — display size or sizes of a page of
    memory, 1073
  pargs — print process arguments, environment
    variables, or auxiliary vector, 1074
  Pascal
    create a tags file for use with ex and vi —
    ctags, 252
  passwd — change login password and
  password attributes, 1076
  password
    change in NIS — yppasswd, 1696
  password file
    edit — vipw, 1651
  passwords
    change login password and password
    attributes — passwd, 1076
  paste — merge same lines of several files or
    subsequent lines of one file, 1082
  patch — apply changes to files, 1085
    File Name Determination, 1088
    Patch Application, 1088
    Patch File Format, 1087
  PATH variable — sh, 1378
  patchchk — check path names, 1090
  pathconv — search FMLI criteria for
    filename, 1093
  pathname
    prompts for a pathname — ckpath, 150
    provides error message for pathname —
    errpath, 150
    provides help message for pathname —
    helppath, 150
    validates pathname — valpath, 150
  pattern scanning and processing language —
    nawk, 941
  pax — portable archive interchange, 1095
    Modes of Operations, 1095
    Standard Error, 1101
    Standard Output, 1100
  pcat — compress files, 1070
  pcred — proc tools, 1203
  pdp11 — get processor type truth value, 787
  performance monitoring
    display call-graph profile data — gprof, 508
    resource usage for a command —
    rusage, 1289
    time a command; report process data and
    system activity — timex, 1537
  perl — Practical Extraction and Report
    Language, 1104
  pfsh — execute a command in a profile, 1111
  pfexec — execute a command in a profile, 1111
  pfiles — proc tools, 1203
  pfksh — execute a command in a profile, 1111
  pfsh — execute a command in a profile, 1111
  pg — files perusal filter for CRTs, 1112

Index 1717
pgrep — find processes by name and other attributes, 1117
pkginfo — display software package information, 1121
pkgmk — produce an installable package, 1123
pkgparam — display package parameter values, 1126
pkgproto — generate prototype file entries for input to pkgmk command, 1128
pkgtrans — translate package format, 1130
pkill — signal processes by name and other attributes, 1117
pldd — proc tools, 1203
plimit — get or set the resource limits of running processes, 1132
plot — graphics filters for plotters, 1134
plotters
  graphics filters — plot, 1134
  graphics filters — tplot
  graphics filters, 1559
pmap — display information about the address space of a process, 1136
popd — shell built-in functions to change the current working directory, 115
portable archive interchange — pax, 1095
postplot — PostScript translator for plot(4B) graphics files, 1153
postdaisy — PostScript translator for Diablo 630 daisy-wheel files, 1143
postdmd — PostScript translator for DMD bitmap files, 1145
postio — serial interface for PostScript printers, 1147
postmd — matrix display program for PostScript printers, 1150
postprint — PostScript translator for text files, 1155
postprocessors
  troff for PostScript printers — dpost, 294
postreverse — reverse the page order in a PostScript file, 1157
PostScript
  matrix display program — postmd, 1150
  prepends host resident PostScript fonts to files — download, 292
  reverse the page order in a PostScript file — postreverse, 1157
PostScript (continued)
  serial interface — postio, 1147
  translator for Diablo 630 daisy-wheel files — postdaisy, 1143
  translator for DMD bitmap files — postdmd, 1145
  translator for plot(4B) graphics files — postplot, 1153
  translator for Tektronix 4014 files — posttek, 1159
  translator for text files — postprint, 1155
troff postprocessor for PostScript printers — dpost, 294
PostScript translator for Diablo 630 daisy-wheel files — postdaisy, 1143
PostScript translator for MD bitmap files — postdmd, 1145
PostScript translator for Tektronix 4014 files — posttek, 1159
PostScript translator for text files — postprint, 1155
posttek — PostScript translator for Tektronix 4014 files, 1159
pr — print files, 1164
Practical Extraction and Report Language — perl, 1104
praliases — display system mail aliases, 1168
prctl — get or set the resource controls of running processes, tasks, and projects, 1169
preap — force a defunct process to be reaped by its parent, 1172
prex — control tracing and manipulate probe points in a process or the kernel, 1174
prime factors
  obtain for a number — factor, 377
print — shell built-in function to output characters to the screen or window, 1185
print
  formatted output — printf, 1187
  print files — pr, 1164
print process arguments, environment variables, or auxiliary vector — params, 1074
print authorizations granted to a user — auths, 83
print execution profiles for a user — profiles, 1210
print files — lpr, 761
print files
  prepends host resident PostScript fonts to files — download, 292
print project membership of user — projects, 1212
print roles granted to a user — roles, 1270
print services
  print information about the status — lpstat, 767
printenv — display environment variables, 1186
printers
  cancel requests — cancel, 109
  control — lpc, 755
  display queue — lpq, 759
  print information about the status — lpstat, 767
  remove jobs from queue — lprm, 765
  send requests — lp, 749
  test — lptest, 771
printers, LP
  — disable, 337
  — enable, 337
printf — print formatted output, 1187
proc tools
  — pcred, 1203
  — pfiles, 1203
  — pflags, 1203
  — pdda, 1203
  — prun, 1203
  — psig, 1203
  — pstack, 1203
  — pstop, 1203
  — ptime, 1203
  — ptree, 1203
  — pwait, 1203
  — pwdx, 1203
process, running
  change priority — renice, 1249
process accounting
  search and print files — acctcom, 28
  time a command; report process data and system activity — timex, 1537
process scheduler
  display or set scheduling parameters of specified process(es) — priocntl, 1192
process status
  report — ps, 1213
processes
  display status — ps, 1222
  get core images of running processes — gcore, 453
  terminate a process by default — kill, 569
processors
  display type — mach, 786
  prof — display profile data, 1206
profile
  display call-graph — gprof, 508
profiles — print execution profiles for a user, 1210
programming languages
  analyze and disperse compiler error messages — error, 346
  C compiler — cc, 113
  C preprocessor — cpp, 196
  C program verifier — lint, 703
  extract strings from C code — xstr, 1687
  formats program in nice style using troff — vgrind, 1637
programming tools
  arbitrary precision arithmetic language — bc, 94
  assembler — as, 60
  create a tags file for use with ex and vi — ctags, 252
  create C error messages — mkstr, 912
  debugger — adb, 31
  display call-graph profile data — gprof, 508
  dump selected parts of an object file — dump, 302
  find printable strings in an object or binary file — strings, 1453
  — install, 542
  link editor — ld, 648
  link-editor for object files — ld, 636
  macro processor — m4, 781
  make — build programs, 818
  object code disassembler — dis, 286
  print name list of an object file — nm, 1049
  print section sizes in bytes of object files — size, 1400
  regular expression compile — regcmp, 1244
  resolve and remove ifdef’d lines from C program source — unifdef, 1602
  resource usage for a command — rusage, 1289
programming tools (continued)
  RPC protocol compiler — rpcgen, 1272
  Source Code Control System — sccs, 1300
  strip symbol table, debugging and line
    number information from an object file —
      strip, 1455
  touch — update last modified date of
    file, 1558
  projects — print project membership of
    user, 1212
  prun — proc tools, 1203
  ps — display process status, 1222
  PS1 variable — sh, 1379
  PS2 variable — sh, 1379
  psig — proc tools, 1203
  pstack — proc tools, 1203
  pstop — proc tools, 1203
  ptime — proc tools, 1203
  ptree — proc tools, 1203
  pushd — shell built-in functions to change the
    current working directory, 115
  pvs — display the internal versioning
    information of dynamic objects, 1225
  pwait — proc tools, 1203
  pwd — proc tools, 1203
  pwdx — print working directory name, 1228
  read — shell built-in function to receive from
    standard input (keyboard), 1237
  readfile, longline — (FMLI utility) reads file,
    gets longest line, 1240
  readonly — shell built-in function to protect the
    value of the given variable from
    reassignment, 1241
  reboot system without checking disks —
    fastboot, 378
  red — text editor, 312
  refer — expand and insert references from a
    bibliographic database, 1242
  regcmp — regular expression compile, 1244
  regenerate groups of programs —
    sysV-make, 1476
  regenerate programs — make, 818
  regex — (FMLI utility) match patterns against a
    string, 1246
  registration, 1406
  rehash — shell built-in functions to evaluate the
    internal hash table of the contents of
    directories, 523
  reinit — (FMLI utility) changes the descriptors
    in the initialization file, 1248
  relational database
    — join, 559
  reminder services
    — calendar, 107
  mail notifier — biff, 103
  remote shell — rsh, 1278
  remote system
    connect — tip, 1539
    connect to — cu, 255
    execute a command on a remote system, but
      with the local environment — on, 1067
    file copy — rcp, 1230
    file distribution — rdist, 1232
    remote login — rlogin, 1253
    shell — rsh, 1278
    show status — rup, 1286
    spawn login — ct, 250
    system to system command execution —
      uux, 1626
    transfer files to and from — tftp, 1530
    who is logged in on remote machines —
      rusers, 1291
  removable rewritable media format utility —
    rmformat, 1260
  rename the binding of an FNS name —
    fnrename, 429
renice — alter priority of running
    processes, 1249
report on the calls to a specific procedure —
    whocalls, 1675
report or filter out repeated lines in a file —
    uniq, 1604
reset — (FLMI utility) changes the entry in a
    field of a form to its default value, 1252
reset terminal bits — reset, 1583
return — shell built-in functions to enable the
    execution of the shell to advance beyond its
    sequence of steps, 361
reverse page order
    PostScript file — postreverse, 1157
reverse the page order in a PostScript file —
    postreverse, 1157
rksh — KornShell, restricted command and
    programming language, 581
login — remote login, 1253
rm — remove files, 1256
rmrdir — remove directories, 1256
rmformat — removable rewritable media
    format utility, 1260
roffbib — format and print bibliographic
    database, 1268
roles — print roles granted to a user, 1270
RPC
    display host status of remote machines —
        rup, 1287
    display status of network hosts — rup, 1286
    protocol compiler — rpcgen, 1272
RPC, secure
    decrypt and store secret key with keyserv —
        keylogin, 566
    delete stored secret key with keyserv —
        keylogout, 568
RPC Language
    RPC protocol compiler — rpcgen, 1272
    rpcgen — RPC protocol compiler, 1272
    rpm2cpio — convert Red Hat Package (RPM) to
        cpio archive, 1277
rsh — remote shell, 1278
run — (FMLI utility) runs a program, 1281
run a command immune to hangups —
    nohup, 1054
runat — execute command in extended
    attribute name space, 1283
runtime linker for dynamic objects —
    ld.so.1, 677
rup — display status of network hosts (RPC
    version), 1286
rup — display status of network hosts, 1287
rusage — display status of local hosts, 1288
rusers — who is logged in on remote
    machines, 1291
rwho — who is logged in on local
    machines, 1292

S
sag — system activity graph, 1293
sar — system activity reporter, 1295
SCCS
    extract SCCS version information from a file
        — what, 1666
    sccs-admin — create and administer SCCS
        history files, 1310
    sccs-cdc — change the delta commentary of an
        SCCS delta, 1314
    sccs-comb — combine deltas, 1316
    sccs — Source Code Control System, 1300
    SCCS commands
        admin — create and administer SCCS history
            files, 1310
        cdc — change the delta commentary of an
            SCCS delta, 1314
        comb — combine deltas, 1316
        delta — change the delta commentary of an
            SCCS delta, 1318
        get — retrieve a version of an SCCS
            file, 1321
        help — help regarding SCCS error or
            warning messages, 1327
        prt — display delta table information from
            an SCCS file, 1332
        rmdel — remove a delta from an SCCS
            file, 1335
        sact — show editing activity status of an
            SCCS file, 1336
        sccs-prs — display selected portions of an
            SCCS file, 1337
        sccsdiff — compare versions of SCCS
            file, 1337

Index 1721
SCCS commands (continued)
unget — unget SCCS file, 1338
val — validate SCCS file, 1339
sccs-delta — change the delta commentary of an
SCCS delta, 1318
SCCS delta
change commentary — sccs-cdc, 1314
combine — sccs-comb, 1316
create — delta, 1318
remove — rmdel, 1335
SCCS delta table
print form an SCCS file — sccs-prt, 1332
SCCS files
compare versions — sccs-sccsdiff, 1337
retrieve a version of a file — sccs-get, 1321
show editing activity status —
sccs-sact, 1336
undo a previous get of an SCCS file —
sccs-unget, 1338
validate — sccs-val, 1339
sccs-get — retrieve a version of an SCCS
file, 1321
sccs-help — help regarding SCCS error or
warning messages, 1327
SCCS help
regarding SCCS error or warning messages
— sccs-help, 1327
SCCS history
display selected portions — sccs-prs, 1328
SCCS history files
create and administer — sccs-admin, 1310
sccs-prs — display selected portions of an SCCS
history, 1328
sccs-prt — display delta table information from
an SCCS file, 1332
sccs-rmdel — remove delta from SCCS
file, 1335
sccs-sact — show editing activity status of an
SCCS file, 1336
sccs-sccsdiff — compare versions of SCCS
file, 1337
sccs-unget — unget SCCS file, 1338
sccs-val — validate SCCS file, 1339
scp — secure copy (remote file copy
program), 1341
screen-oriented editor — vi, 1641
script — make script of terminal session, 1343
sdiff — print differences between two files
side-by-side, 1344
search for FNS objects with specified attributes
— fnsearch, 430
secure copy (remote file copy program) —
scp, 1341
secure file transfer program — sftp, 1372
Secure Shell proxy for HTTP —
ssh-http-proxy-connect, 1443
Secure Shell proxy for SOCKS5 —
ssh-socks5-proxy-connect, 1448
sed — stream editor, 1346, 1353
Functions, 1354
sed Addresses, 1347
sed Editing Commands, 1347
sed Regular Expressions, 1347
sed Scripts, 1353
select or reject lines common to two files —
comm, 172
serial interface for Postscript printers —
postio, 1147
set — shell built-in functions to determine the
characteristics for environmental variables of
the current shell and its descendents, 1359
set, unset — (FLMI utility) set and unset local or
global environment variables, 1364
set environment for command invocation —
env, 339
set or get limitations on the system resources
available to the current shell and its
descendents
— limit, 698
— ulimit, 698
— unlimit, 698
set process group ID — setpgrp, 1371
setcolor — (FMLI utility) redefine or create a
color, 1366
setenv — shell built-in functions to determine
the characteristics for environmental
variables of the current shell and its
descendents, 1359
setfacl — modify the Access Control List (ACL)
for a file or files, 1367
acl_entries Syntax, 1367
setpgrp — set process group ID, 1371
settime — change file access and modification
times, 1555
sftp — secure file transfer program, 1372
sh — the standard shell command interpreter, 1375
SHACCT variable — sh, 1379
shared object for setting preferred page size — mpss.so.1, 926
shell — (FMLI utility) run a command using shell, 1393
SHELL variable — sh, 1379
shell
Korn shell — ksh, 581
restricted Korn shell — rksh, 581
shell command interpreter builtin-functions
— alias, 34
— bg, 552
— break, 104
— cd, 115
— chdir, 115
— continue, 104
— dirs, 115
— eval, 359
— exit, 361
— fc, 527
— fg, 552
— getopts, 497
— glob, 507
— hash, 523
— hashstat, 523
— history, 527
— jobs, 552
— kill, 569
— let, 685
— logout, 745
— newgrp, 975
— notify, 552
— onintr, 1570
— popd, 115
— print, 1185
— pushd, 115
— read, 1237
— readonly, 1241
— rehash, 523
— return, 361
— set, 1359
— setenv, 1359
— shift, 1398
— source, 359
— stop, 552
— suspend, 1474
shell command interpreter builtin-functions
(continued)
— times, 1536
— trap, 1570
— typeset, 1592
— umask, 1596
— unalias, 34
— unhash, 523
— unset, 1359
— unsetenv, 1359
— wait, 1661
— whence, 1592
shell programming
echo arguments — echo, 306
read one line from standard input and write to standard output — line, 702
shell scripts
display size or sizes of a page of memory — pagesize, 1073
provide truth values — true, false, 1575
shell variables, in Bourne shell, 1378
shells
C shell — csh, 221
remote — rsh, 1278
the job control shell command interpreter — jsh, 1375
the standard shell command interpreter — sh, 1375
shift — shell built-in function to traverse either a shell’s argument list or a list of field-separated words, 1398
show codeset table for the current locale — dumpcs, 305
show current number of users in each FTP Server class — ftpcount, 451
show current process information for each FTP Server user — ftpwho, 452
shutdown — shut down multiuser operation, 1399
sign on to the system — login, 737
Simple Mail Transfer Protocol connection to remote mailserver — mconnect, 859
size — print section sizes in bytes of object files, 1400
sleep — suspend execution for an interval, 1402
Index 1723
smart2cfg — Compaq Smart-2 EISA/PCI and Smart-2SL PCI Array Controller ioctl utility, 1403
SMPT, See Simple Mail Transfer Protocol
soelim — eliminate .so’s from nroff input, 1405
software package
display information — pkginfo, 1121
display parameter values — pkgparam, 1126
generate prototype file entries for input to pkgmk command — pkgproto, 1128
produce an installable package — pkgmk, 1123
translate package format — pkgtrans, 1130
Solaris user registration — solregis, 1406
solregis — Solaris user registration, 1406
sort, topological
items mentioned in input — tsort, 1588
sort — sort and/or merge files, 1409
sortbib — sort bibliographic database, 1416
sotruss — trace shared library procedure calls, 1418
source — shell built-in functions to execute other commands, 359
Source Code Control System, See SCCS
source files
locate — whereis, 1668
sparc — get processor type truth value, 787
spell — check spelling, 1420
spline — interpolate smooth curve, 1423
split files based on context — csplit, 247
split — split a file into pieces, 1424
srchtxt — display contents of, or search for a text string in, message data bases, 1426
strchg — change stream configuration, 1450
strconf — query stream configuration, 1450
stream editor — sed, 1346, 1353
STREAMS
change or query stream configuration — strchg, strconf, 1450
string
prompt for defined string answer — ckstr, 156
provide an error message for defined string answer — erstr, 156
provide an help message for defined string answer — helpstr, 156
validate a defined string answer — valstr, 156
strings — find printable strings in object or binary file, 1453
strip — strip symbol table, debugging and line number information from an object file, 1455
stty command, 1465
stty — set the options for a terminal, 1457
sum — print checksum and block count for a file, 1472
sum — sum and count blocks in file, 1473
sun — get processor type truth value, 787
SunOS/BSD Source Compatibility Package
— stty, 1465
SunOS/BSD Source Compatibility Package commands
— arch, 59
— basename, 93
— biff, 103
— cc, 113
— chown, 137
— df, 275
— du, 300
— echo, 310
— expr, 369
— fastboot, 378
— file, 387
— from, 439
— groups, 521
— grpck, 522

statistics
— collected by sendmail — mailstats, 795
— stop — shell built-in functions to control process execution, 552
strchg — change stream configuration, 1450
strconf — query stream configuration, 1450
SunOS/BSD Source Compatibility Package commands (continued)
— hostid, 536
— hostname, 537
— install, 542
— ld, 648
— lint, 703
— ln, 722
— logger, 735
— lpc, 755
— lpq, 759
— lpr, 761
— lprm, 765
— lptest, 771
— ls, 778
— mach, 786
— mkstr, 912
— pagesize, 1073
— plot, 1134
— printenv, 1186
— ps, 1222
— rusage, 1289
— shutdown, 1399
— sum, 1473
— test, 1526
— tr, 1569
— tset, 1583
— users, 1611
— vipw, 1651
— whereis, 1668
— whoami, 1674

provide SunOS compatibility for Solaris
mailbox format — mailcompat, 790
suspend — shell built-in function to halt the
 current shell, 1474
suspend execution of command
— sleep, 1402
symorder — update symbol table
 ordering, 1475
synchronize files and directories —
 filesync, 389
system to system copy — uucp, 1612
system activity
 graphical representation — sag, 1293
 reporter — sar, 1295
time a command; report process data and
 system activity — timex, 1537

system administration
— install, 542
system call and signals
 trace — truss, 1576
system log
 add entries — logger, 733
system name
 print — uname, 1599
system to system command execution —
 uux, 1626
system uptime
 display — uptime, 1610
sysV-make — maintain, update, and regenerate
 groups of programs, 1476

T
TAB characters
 expand to SPACE characters, and vice versa
 — expand, unexpand, 363
tables
 format for nroff or troff — tbl, 1504
tabs — set tabs on a terminal, 1483
tail — display last part of file, 1487
talk — talk to another user, 1490
tape
 backspace files — mt, 935
 backspace records — mt, 935
 erase — mt, 935
 forward space files — mt, 935
 forward space records — mt, 935
 get unit status — mt, 935
 place unit off-line — mt, 935
 retention — mt, 935
 rewind — mt, 935
 skip backward files — mt, 935
 skip backward records — mt, 935
 skip forward files — mt, 935
 skip forward records — mt, 935
 write EOF mark on — mt, 935
tape, magnetic
 copy, blocking preserved — tcopy, 1506
 manipulate — mt, 935
 scan — tcopy, 1506
tape archives
 create — tar, 1493
tar — create tape archives, and add or extract files, 1493
tbl — format tables for nroff or troff, 1504
        tbl
        remove nroff, troff, tbl and eqn constructs —
        deroff, 274
tcopy — copy a magnetic tape, 1506
tee — replicate the standard output, 1507
telnet — user interface to a remote system using the TELNET protocol, 1508
        TELNET protocol
        user interface to a remote system using the TELNET protocol — telnet, 1508
terminal
        set options — stty, 1457
        set tabs — tabs, 1483
terminal screen
        — clear, 167
terminal session
        make script— script, 1343
terminals
        get name — tty, 1590
        initialize a terminal or query terminfo database — tput, 1560
        reset bits — reset, 1583
        set characteristics — stset, 1483
        set characteristics — stty, 1465
        terminate a process by default — kill, 569
terminfo database
        initialize a terminal or query terminfo database — tput, 1560
test — evaluate condition(s), 1518
        test — (FMLI utility) evaluates the expression
        expression, 1528
test — condition evaluation, 1526
text editing
        screen-oriented (visual) display editor based on ex — vi, 1641
        sed — stream editor, 1353
        stream editor — sed, 1346
text editor
        — ed, 312
        — edit, 324
        — ex, 350
text files
        browse or page through a text file — more, page, 914
        change format — newform, 972
        text formatter
        format documents for display or line-printer
        — nroff, 1058
        text processing utilities
        check spelling — spell, 1420
        concatenate and display files — cat, 111
        display last part of file — tail, 1487
        pattern scanning and processing language — awk, 85
        search a file for a pattern — grep, 515
        search a file for a pattern using full regular expressions — egrep, 328
        search file for fixed-character string — fgrep, 383
        sort and/or merge files — sort, 1409
        split a file into pieces — split, 1424
        translate characters — tr, 1564, 1569
        underline text — ul, 1595
text retrieval tools
        create message files for use by gettxt — mkmsgs, 910
        retrieve text string from message database — gettxt, 505
text to PDL (Printer Description Language)
        pretty print filter — mp, 920
tftp — trivial file transfer program, 1530
        the Network Cache and Accelerator (NCA) —
        NCA, 962
        the Network Cache and Accelerator (NCA) — nca, 962
tilde escape commands for mail
        — mailx, 808
time — time a simple command, 1533
        time
        prompts for time — cktime, 161
        provides error message for time —
        errtime, 161
        provides help message for time —
        helptime, 161
        validates time — valtime, 161
        time a simple command — time, 1533
        timed event services
        display the jobs queued to run at specified
times — atq, 72
        reminder service — calendar, 107
        remove jobs spooled by at or batch —
        atrm, 73
        user crontab file — crontab, 216
timemanp — frontends to the mp Text to PDL (Printer Description Language) pretty print filter, 791
times — shell built-in function to report time usages of the current shell, 1536
timesysp — frontends to the mp Text to PDL (Printer Description Language) pretty print filter, 791
timex — time a command; report process data and system activity, 1537
tip — connect to remote system, 1539
tnfdump — convert binary TNF file to ASCII, 1548
tnfxtract — extract kernel probes output into a trace file, 1553
touch — change file access and modification times, 1555, 1591
    settime, 1556
    touch, 1555
touch — update last modified date of file, 1558
tplot — graphics filters for plotters, 1559
    tput — initialize a terminal or query terminfo database, 1560
    tr — translate characters, 1564, 1569
    trace function calls
        trace application function calls to Solaris shared libraries — apptrace, 48
        trace shared library procedure calls — sottrss, 1418
        translate characters — tr, 1564, 1569
        translates exportfs options to share/unshare commands — exportfs, 365
        trap — shell built-in functions to respond to (hardware) signals, 1570
    Trivial File Transfer Protocol, See TFTP
troff — typeset or format documents, 1572
troff utilities
    check nroff and troff files — checknr, 124
    eliminate .so’s from nroff input — soelim, 1405
    filters reverse line-feeds from two-column nroff text — col, 170
    format tables — tbl, 1504
    formats program code — vgrind, 1637
    postprocessor for PostScript printers — dpost, 294
    remove nroff, troff, tbl and eqn constructs — deroff, 274
troff — provide truth values, 1575
truss — trace system calls and signals, 1576
tset — set terminal characteristics, 1583
tsort — topological sort of items mentioned in input, 1588
ttl — time to live value, nischttl, 1007
    tty, set characteristics — stty, 1465
    tty, set characteristics — stset, 1583
    tty
        set options — stty, 1457
    tty — get the name of the terminal, 1590
typeset — shell built-in functions to set/get attributes and values for shell variables and functions, 1592
    typeset documents — troff, 1572

U
    u370 — get processor type truth value, 787
    u3b — get processor type truth value, 787
    u3b15 — get processor type truth value, 787
    u3b2 — get processor type truth value, 787
    u3b5 — get processor type truth value, 787
    ucblinks — adds /dev entries to give SunOS 4.x compatible names to SunOS 5.x
        devices, 1594
    ul — underline text, 1595
    ulimit — set or get limitations on the system resources available to the current shell and its descendents, 698
    umask — shell built-in function to restrict read/write/execute permissions, 1596
    unalias — shell built-in functions to create your own pseudonym or shorthand for a command or series of commands, 34
    uname — print name of current system, 1599
    unbind the reference from an FNS name — fnunbind, 436
    uncompress — uncompress files, 177
    underline text — ul, 1595
    unexpand — unexpand SPACE characters to TAB characters, 363
    unhash — shell built-in functions to evaluate the internal hash table of the contents of directories, 523
    unifdef — resolve and remove ifdef’ed lines from C program source, 1602
uniq — report or filter out repeated lines in a file, 1604
units — converts quantities expressed in standard scales to other scales, 1606
UNIX
   convert text file from DOS format to ISO format — dos2unix, 290
UNIX-to-UNIX commands
   uucp — uucp, 1612
   uulog — uucp, 1612
   uuname — uucp, 1612
unix2dos — convert text file from ISO format to DOS format, 1608
unlimit — set or get limitations on the system resources available to the current shell and its descendents, 698
unpack — expand compressed files, 1070
unset — shell built-in functions to determine the characteristics for environmental variables of the current shell and its descendents, 1359
unsetenv — shell built-in functions to determine the characteristics for environmental variables of the current shell and its descendents, 1359
update and examine attributes associated with FNS named object — fnattr, 421
update groups of programs —
   sysV-make, 1476
update last modified date of file — touch, 1558
update programs — make, 818
uptime — show how long the system has been up, 1610
user ID
   change user IDs of files — chown, 137
user IDs
   display a list of all valid user names — dispuid, 289
   prompts for user ID — ckuid, 163
   provides error message for user ID — erruid, 163
   provides help message for user ID — helpuid, 163
   validates user ID — valuid, 163
users
   display effective name — whoami, 1674
   display group membership — groups, 520
users (continued)
   display information about local and remote users — finger, 403
   get the name of the user running the process — logname, 744
   list user login information — listusers, 707
   talk to another user — talk, 1490
   who is logged in, and what are they doing — w, 1659
   who is logged in on local machines — rwho, 1292
   who is logged in on remote machines — rusers, 1291
   who is on the system — who, 1671
   write to another user — write, 1677
users, network
   Internet user name directory service — whois, 1676
   users — display users on system, 1611
   uucp — UNIX-to-UNIX copy, 1612
   uucp
      log — uulog, 1612
      uucp status inquiry — uustat, 1619
      uudecode — decode binary file, 1616
      uuencode — encode binary file, 1616
      uuglist — print list of service grades available, 1618
      uulog — UUCP log, 1612
      uuname — UUCP list of names, 1612
      uustat — uucp status inquiry, 1619
   uux — system to system command execution, 1626
V
   vacation — automatic mail replies, 1630
   vax — get processor type truth value, 787
version control
   — vc, 1633
vgrind — formats program in nice style using troff, 1637
vi — screen-oriented (visual) display editor based on ex, 1641
vipw — edit password file, 1651
volcancel — cancel user’s request for removable media that is not currently in drive, 1652
volcheck — check for media in a drive, 1653
volmissing — notify user that volume requested is not in the CD-ROM or floppy drive, 1655
volrmount — call rmmount to mount or unmount media, 1656
Volume Management
cancel user’s request for removable media that is not currently in drive —
volcancel, 1652
check for media in a drive —
volcheck, 1653
missing volume notification — volmissing, 1655
vsiq — synchronize a co-process with the controlling FMLI application, 1658

W
w — display information about currently logged-in users, 1659
w — who is logged in, and what are they doing, 1659
wait — shell built-in function to wait for other jobs or processes, 1661
wc — display a count of lines, words and characters in a file, 1664
what — extract SCCS version information from a file, 1666
whatis — describe command, 1667
whence — shell built-in functions to set/get attributes and values for shell variables and functions, 1592
whereis — locate the binary, source and manual page files for a command, 1668
which — locate a command; display its pathname or alias, 1670
who is logged in — w, 1659
who — who is on the system, 1671
whoami — display effective user name, 1674
whocalls — report on the calls to a specific procedure, 1675
whois — Internet user name directory service, 1676
write — write to another user, 1677
write file checksums and sizes — cksum, 159

X
xargs — construct argument lists and invoke utility, 1680
xgettext — extract gettext call strings, 1685
xstr — extract strings from C code, 1687

Y
yacc — yet another compiler-compiler, 1689
yacc create a tags file for use with ex and vi — ctags, 252
yes — generate repetitive affirmative output, 1693
yes/no answer
  prompts for yes/no answer — ckyorn, 165
  provides error message for yes/no answer — erryorn, 165
  provides help message for yes/no answer — helpyorn, 165
  validates yes/no answer — valyorn, 165
yet another compiler-compiler — yacc, 1689
ypcat — print values in a NIS database, 1694
ypmatch — print the value of one or more keys from a NIS map, 1695
yppasswd — change your network password in the NIS database, 1696
ypwhich — return name of NIS server or map master, 1697

Z
zcat — displays uncompressed files but leaves compressed files intact, 177