man pages section 4: File Formats
| **shells(4)** | 439 |
| **slp.conf(4)** | 440 |
| **slpd.reg(4)** | 448 |
| **sock2path(4)** | 450 |
| **space(4)** | 451 |
| **sulog(4)** | 452 |
| **sysbus(4)** | 453 |
| **sysidcfg(4)** | 456 |
| **syslog.conf(4)** | 461 |
| **system(4)** | 464 |
| **telnetrc(4)** | 468 |
| **term(4)** | 469 |
| **terminfo(4)** | 472 |
| **TIMEZONE(4)** | 526 |
| **timezone(4)** | 527 |
| **tnf_kernel_probes(4)** | 528 |
| **ts_dptbl(4)** | 535 |
| **ttydefs(4)** | 542 |
| **ttysrch(4)** | 543 |
| **ufsdump(4)** | 545 |
| **updaters(4)** | 551 |
| **user_attr(4)** | 552 |
| **utmp(4)** | 555 |
| **utmpx(4)** | 556 |
| **vfsstab(4)** | 557 |
| **vold.conf(4)** | 558 |
| **warn.conf(4)** | 562 |
| **ypfiles(4)** | 563 |
| **zoneinfo(4)** | 565 |

**Index** 567
Preface

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

Overview

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

- Section 1 describes, in alphabetical order, commands available with the operating system.
- Section 1M describes, in alphabetical order, commands that are used chiefly for system maintenance and administration purposes.
- Section 2 describes all of the system calls. Most of these calls have one or more error returns. An error condition is indicated by an otherwise impossible returned value.
- Section 3 describes functions found in various libraries, other than those functions that directly invoke UNIX system primitives, which are described in Section 2.
- Section 4 outlines the formats of various files. The C structure declarations for the file formats are given where applicable.
- Section 5 contains miscellaneous documentation such as character-set tables.
- Section 6 contains available games and demos.
- Section 7 describes various special files that refer to specific hardware peripherals and device drivers. STREAMS software drivers, modules and the STREAMS-generic set of system calls are also described.
Section 9 provides reference information needed to write device drivers in the kernel environment. It describes two device driver interface specifications: the Device Driver Interface (DDI) and the Driver/Kernel Interface (DKI).

Section 9E describes the DDI/DKI, DDI-only, and DKI-only entry-point routines a developer can include in a device driver.

Section 9F describes the kernel functions available for use by device drivers.

Section 9S describes the data structures used by drivers to share information between the driver and the kernel.

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

NAME

This section gives the names of the commands or functions documented, followed by a brief description of what they do.

SYNOPSIS

This section shows the syntax of commands or functions. When a command or file does not exist in the standard path, its full path name is shown. Options and arguments are alphabetized, with single letter arguments first, and options with arguments next, unless a different argument order is required.

The following special characters are used in this section:

[ ] Brackets. The option or argument enclosed in these brackets is optional. If the brackets are omitted, the argument must be specified.

... Ellipses. Several values can be provided for the previous argument, or the previous argument can be specified multiple times, for example, "filename ...".

| Separator. Only one of the arguments separated by this character can be specified at a time.

{ } Braces. The options and/or arguments enclosed within braces are interdependent, such that everything enclosed must be treated as a unit.
PROTOCOL
This section occurs only in subsection 3R to indicate the protocol description file.

DESCRIPTION
This section defines the functionality and behavior of the service. Thus it describes concisely what the command does. It does not discuss OPTIONS or cite EXAMPLES. Interactive commands, subcommands, requests, macros, and functions are described under USAGE.

IOCTL
This section appears on pages in Section 7 only. Only the device class that supplies appropriate parameters to the 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 mtic(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
<table>
<thead>
<tr>
<th>NAME</th>
<th>Intro – introduction to file formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCRIPTION</td>
<td>This section outlines the formats of various files. The C structure declarations for the file formats are given where applicable. Usually, the headers containing these structure declarations can be found in the directories <code>/usr/include</code> or <code>/usr/include/sys</code>. For inclusion in C language programs, however, the syntax <code>#include &lt;filename.h&gt;</code> or <code>#include &lt;sys/filename.h&gt;</code> should be used. Because the operating system now allows the existence of multiple file system types, there are several instances of multiple manual pages with the same name. These pages all display the name of the FSType to which they pertain, in the form <code>name_fstype</code> at the top of the page. For example, <code>fs_ufs(4)</code></td>
</tr>
</tbody>
</table>
File Formats
### NAME
admin – installation defaults file

### DESCRIPTION
admin is a generic name for an ASCII file that defines default installation actions by assigning values to installation parameters. For example, it allows administrators to define how to proceed when the package being installed already exists on the system.

/var/sadm/install/admin/default is the default admin file delivered with this release. The default file is not writable, so to assign values different from this file, create a new admin file. There are no naming restrictions for admin files. Name the file when installing a package with the -a option of pkgadd(1M). If the -a option is not used, the default admin file is used.

Each entry in the admin file is a line that establishes the value of a parameter in the following form:

```
param=value
```

Eleven parameters can be defined in an admin file, but it is not required to assign values to all eleven parameters. If a value is not assigned, pkgadd(1M) asks the installer how to proceed.

The eleven parameters and their possible values are shown below except as noted. They may be specified in any order. Any of these parameters (except the mail parameter) can be assigned the value ask, which means that if the situation occurs the installer is notified and asked to supply instructions at that time (see NOTES).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>basedir</td>
<td>Indicates the base directory where relocatable packages are to be installed. If there is no basedir entry in the file, the installer will be prompted for a path name, as if the file contained the entry basedir=ask. This parameter can also be set to default (entry is basedir=default). In this instance, the package is installed into the base directory specified by the BASEDIR parameter in the pkginfo(4) file.</td>
</tr>
<tr>
<td>mail</td>
<td>Defines a list of users to whom mail should be sent following installation of a package. If the list is empty, no mail is sent. If the parameter is not present in the admin file, the default value of root is used. The ask value cannot be used with this parameter.</td>
</tr>
<tr>
<td>runlevel</td>
<td>Indicates resolution if the run level is not correct for the installation or removal of a package. Options are:</td>
</tr>
<tr>
<td></td>
<td>nocheck</td>
</tr>
<tr>
<td></td>
<td>quit</td>
</tr>
<tr>
<td>conflict</td>
<td>Specifies what to do if an installation expects to overwrite a previously installed file, thus creating a conflict between packages. Options are:</td>
</tr>
</tbody>
</table>
### setuid
Checks for executables which will have setuid or setgid bits enabled after installation. Options are:

- **nocheck**: Do not check for setuid executables.
- **quit**: Abort installation if setuid processes are detected.
- **nochange**: Override installation of setuid processes; processes will be installed without setuid bits enabled.

### action
Determines if action scripts provided by package developers contain possible security impact. Options are:

- **nocheck**: Ignore security impact of action scripts.
- **quit**: Abort installation if action scripts may have a negative security impact.

### partial
Checks to see if a version of the package is already partially installed on the system. Options are:

- **nocheck**: Do not check for a partially installed package.
- **quit**: Abort installation if a partially installed package exists.

### instance
Determines how to handle installation if a previous version of the package (including a partially installed instance) already exists. Options are:

- **quit**: Exit without installing if an instance of the package already exists (does not overwrite existing packages).
- **overwrite**: Overwrite an existing package if only one instance exists. If there is more than one instance, but only one has the same architecture, it overwrites that instance. Otherwise, the installer is prompted with existing instances and asked which to overwrite.
- **unique**: Do not overwrite an existing instance of a package. Instead, a new instance of the package is created. The new instance will be
assigned the next available instance identifier.

**idepend**  Controls resolution if other packages depend on the one to be installed. Options are:

- **nocheck**  Do not check package dependencies.
- **quit**  Abort installation if package dependencies are not met.

**rdepend**  Controls resolution if other packages depend on the one to be removed. Options are:

- **nocheck**  Do not check package dependencies.
- **quit**  Abort removal if package dependencies are not met.

**space**  Controls resolution if disk space requirements for package are not met. Options are:

- **nocheck**  Do not check space requirements (installation fails if it runs out of space).
- **quit**  Abort installation if space requirements are not met.

**EXAMPLES**

**EXAMPLE 1** Sample of admin file.

Below is a sample admin file.

```
basedir=default
runlevel=quit
conflict=quit
setuid=quit
action=quit
partial=quit
instance=unique
idepend=quit
rdepend=quit
space=quit
```

**SEE ALSO**  pkgadd(1M), pkginfo(4)

**NOTES**  The value `ask` should not be defined in an admin file that will be used for non-interactive installation (since by definition, there is no installer interaction). Doing so causes installation to fail when input is needed.
alias – alias table file of encoding names

SYNOPSIS
/usr/lib/iconv/alias

DESCRIPTION
This file contains the alias table of encoding names for iconv_open(3C).

The format of the alias table is as follows:
"%s %s
", <variant encoding name>, <canonical encoding name>

The string specified for the variant encoding name is case-insensitive. A line beginning
with ‘#’ is treated as a comment.

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

<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
iconv(3C), iconv_close(3C), iconv_open(3C), attributes(5)
NAME aliases, addresses, forward – addresses and aliases for sendmail

SYNOPSIS
/etc/mail/aliases
/etc/mail/aliases.dir
/etc/mail/aliases.pag
~/.forward

DESCRIPTION
These files contain mail addresses or aliases, recognized by sendmail(1M) for the local host:

/etc/passwd  Mail addresses (usernames) of local users.
/etc/mail/aliases  Aliases for the local host, in ASCII format. Root can edit this file to add, update, or delete local mail aliases. Additionally, sendmail(1M) will build the DBM files for /etc/mail/aliases if they are missing, so long as the /etc/mail/aliases* files are owned by root and root has exclusive write permission.

/etc/mail/aliases. {dir, pag}  The aliasing information from /etc/mail/aliases, in binary, dbm format for use by sendmail(1M). The program newaliases(1), which is invoked automatically by sendmail(1M), maintains these files. Also, sendmail(1M) will build the DBM files for /etc/mail/aliases. {dir, pag} if they are missing, so long as /etc/mail/aliases. {dir, pag} is owned by root and root has exclusive write permission.

~/.forward  Addresses to which a user’s mail is forwarded (see Automatic Forwarding).

In addition, the NIS name services aliases map mail.alises, and the NIS+ mail_aliases table, both contain addresses and aliases available for use across the network.

Addresses  As distributed, sendmail(1M) supports the following types of addresses:

Local Usernames  username

Each local username is listed in the local host’s /etc/passwd file.

Local Filenames  pathname

Messages addressed to the absolute pathname of a file are appended to that file.
If the first character of the address is a vertical bar (|), sendmail(1M) pipes the message to the standard input of the command the bar precedes.

username@domain

If domain does not contain any ‘.’ (dots), then it is interpreted as the name of a host in the current domain. Otherwise, the message is passed to a mailhost that determines how to get to the specified domain. Domains are divided into subdomains separated by dots, with the top-level domain on the right.

For example, the full address of John Smith could be:

js@jsmachine.Podunk-U.EDU

if he uses the machine named jsmachine at Podunk University.

...[host!]host!username

These are sometimes mistakenly referred to as “Usenet” addresses. uucp(1C) provides links to numerous sites throughout the world for the remote copying of files.

Other site-specific forms of addressing can be added by customizing the sendmail.cf configuration file. See sendmail(1M) for details. Standard addresses are recommended.

/etc/mail/aliases is formatted as a series of lines of the form

aliasname : address[, address]

aliasname is the name of the alias or alias group, and address is the address of a recipient in the group. Aliases can be nested. That is, an address can be the name of another alias group. Because of the way sendmail(1M) performs mapping from upper-case to lower-case, an address that is the name of another alias group must not contain any upper-case letters.

Lines beginning with white space are treated as continuation lines for the preceding alias. Lines beginning with # are comments.

An alias of the form:

owner-aliasname : address
sendmail directs error-messages resulting from mail to aliasname to address, instead of back to the person who sent the message. sendmail rewrites the SMTP envelope sender to match this, so owner-aliasname should always point to alias-request, and alias-request should point to the owner's actual address:

owner-aliasname: aliasname-request
aliasname-request address

An alias of the form:

aliasname: :include:pathname

with colons as shown, adds the recipients listed in the file pathname to the aliasname alias. This allows a private list to be maintained separately from the aliases file.

The aliases file on the master NIS server is used for the mail.aliases NIS map, which can be made available to every NIS client. The mail.aliases table serves the same purpose on a NIS+ server. Thus, the /etc/mail/aliases* files on the various hosts in a network will one day be obsolete. Domain-wide aliases should ultimately be resolved into usernames on specific hosts. For example, if the following were in the domain-wide alias file:

jsmith:js@jsmachine

eny NIS or NIS+ client could just mail to jsmith and not have to remember the machine and username for John Smith.

If a NIS or NIS+ alias does not resolve to an address with a specific host, then the name of the NIS or NIS+ domain is used. There should be an alias of the domain name for a host in this case.

For example, the alias:

jsmith:root

sends mail on a NIS or NIS+ client to root@podunk-u if the name of the NIS or NIS+ domain is podunk-u.

When an alias (or address) is resolved to the name of a user on the local host, sendmail(1M) checks for a ~/.forward file, owned by the intended recipient, in that user's home directory, and with universal read access. This file can contain one or more addresses or aliases as described above, each of which is sent a copy of the user's mail.
Care must be taken to avoid creating addressing loops in the 
`~/.forward` file. When forwarding mail between machines, be sure that the destination machine does not return the mail to the sender through the operation of any NIS aliases. Otherwise, copies of the message may "bounce." Usually, the solution is to change the NIS alias to direct mail to the proper destination.

A backslash before a username inhibits further aliasing. For instance, to invoke the `vacation` program, user `js` creates a `~/.forward` file that contains the line:

```
\js, "|/usr/ucb/vacation js"
```

so that one copy of the message is sent to the user, and another is piped into the `vacation` program.

### FILES

- `/etc/passwd` password file
- `/etc/nsswitch.conf` name service switch configuration file
- `/etc/mail/aliases` mail aliases file (ascii)
- `/etc/mail/aliases.dir` database of mail aliases (binary)
- `/etc/mail/aliases.pag` database of mail aliases (binary)
- `/etc/mail/sendmail.cf` sendmail configuration file
- `~/.forward` forwarding 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>SUNWsndmr</td>
</tr>
</tbody>
</table>

### SEE ALSO

`newaliases(1), passwd(1), uucp(1C), vacation(1), sendmail(1M), dbm(3UCB), getusershell(3C), passwd(4), shells(4), attributes(5)`

### NOTES

Because of restrictions in `dbm(3UCB)`, a single alias cannot contain more than about 1000 characters. Nested aliases can be used to circumvent this limit.

For aliases which result in piping to a program or concatenating a file, the shell of the controlling user must be allowed. Which shells are and are not allowed are determined by `getusershell(3C)`.
The file name `a.out` is the default output file name from the link editor, `ld(1)`. The link editor will make an `a.out` executable if there were no errors in linking. The output file of the assembler, `as(1)`, also follows the format of the `a.out` file although its default file name is different.

Programs that manipulate ELF files may use the library that `elf(3ELF)` describes. An overview of the file format follows. For more complete information, see the references given below.

<table>
<thead>
<tr>
<th>Linking View</th>
<th>Execution View</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELF header</td>
<td>ELF header</td>
</tr>
<tr>
<td>Program header table</td>
<td>Program header table</td>
</tr>
<tr>
<td>optional</td>
<td></td>
</tr>
<tr>
<td>Section 1</td>
<td>Segment 1</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Section (n)</td>
<td>Segment 2</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Section header table</td>
<td>Section header table</td>
</tr>
<tr>
<td>optional</td>
<td></td>
</tr>
</tbody>
</table>

An ELF header resides at the beginning and holds a “road map” describing the file’s organization. Sections hold the bulk of object file information for the linking view: instructions, data, symbol table, relocation information, and so on. Segments hold the object file information for the program execution view. As shown, a segment may contain one or more sections.

A program header table, if present, tells the system how to create a process image. Files used to build a process image (execute a program) must have a program header table; relocatable files do not need one. A section header table contains information describing the file’s sections. Every section has an entry in the table; each entry gives information such as the section name, the section size, etc. Files used during linking must have a section header table; other object files may or may not have one.

Although the figure shows the program header table immediately after the ELF header, and the section header table following the sections, actual files may differ. Moreover, sections and segments have no specified order. Only the ELF header has a fixed position in the file.
When an a.out file is loaded into memory for execution, three logical segments are set up: the text segment, the data segment (initialized data followed by uninitialized, the latter actually being initialized to all 0's), and a stack. The text segment is not writable by the program; if other processes are executing the same a.out file, the processes will share a single text segment.

The data segment starts at the next maximal page boundary past the last text address. If the system supports more than one page size, the “maximal page” is the largest supported size. When the process image is created, the part of the file holding the end of text and the beginning of data may appear twice. The duplicated chunk of text that appears at the beginning of data is never executed; it is duplicated so that the operating system may bring in pieces of the file in multiples of the actual page size without having to realign the beginning of the data section to a page boundary. Therefore, the first data address is the sum of the next maximal page boundary past the end of text plus the remainder of the last text address divided by the maximal page size. If the last text address is a multiple of the maximal page size, no duplication is necessary. The stack is automatically extended as required. The data segment is extended as requested by the brk(2) system call.

SEE ALSO  as(1), cc(1B), ld(1), brk(2), elf(3ELF)

ANSI C Programmer’s Guide
/* Magic numbers */
#define CMN_ASC 0x070701 /* Cpio Magic Number for -c header */
#define CMN_BIN 070707 /* Cpio Magic Number for Binary header */
#define CMN_BBS 0143561 /* Cpio Magic Number for Byte-Swap header */
#define CMN_CRC 0x070702 /* Cpio Magic Number for CRC header */
#define CMS_ASC "070701" /* Cpio Magic String for -c header */
#define CMS_BIN "070707" /* Cpio Magic String for odc header */
#define CMS_CRC "070702" /* Cpio Magic String for CRC header */
#define CMS_LEN 6 /* Cpio Magic String length */

/* Various header and field lengths */
#define CHRSZ 76 /* -H odc size minus filename field */
#define ASCSZ 110 /* -c and CRC hdr size minus filename field */
#define TARSZ 512 /* TAR hdr size */
#define HNAMLEN 256 /* maximum filename length for binary and odc headers */
#define EXPNLEN 1024 /* maximum filename length for -c and CRC headers */
#define HTIMLEN 2 /* length of modification time field */
#define HSIZLEN 2 /* length of file size field */

/* cpio binary header definition */
struct hdr_cpio {
    short   h_magic,  /* magic number field */
              h_dev;   /* file system of file */
    ushort_t h_ino,   /* inode of file */
    h_mode,   /* modes of file */
    h_uid,    /* uid of file */
    h_gid;    /* gid of file */
    short   h_nlink, /* number of links to file */
    h_rdev,   /* maj/min numbers for special files */
    h_mtime[HTIMLEN], /* modification time of file */
    h_namesize, /* length of filename */
    h_filesize[HSIZLEN]; /* size of file */
    char    h_name[HNAMLEN]; /* filename */
};

/* cpio -H odc header format */
struct c_hdr {
    char c_magic[CMS_LEN],
            c_dev[6],
            c_ino[6],
            c_mode[6],
            c_uid[6],
            c_gid[6],
            c_nlink[6],
            c_rdev[6],
            c_mtime[11],
            c_namesz[6],
            c_filesz[11],
            c_name[HNAMLEN];
};

/* -c and CRC header format */
struct Exp_cpio_hdr {
    char    E_magic[CMS_LEN],
            E_ino[8],
            E_mode[8],
            E_uid[8],
            E_gid[8],

archives(4)
NAME
archives – device header
DESCRIPTION

28
man pages section 4: File Formats • Last Revised 3 Jul 1990
archives(4)

```c
E_nlink[8],
E_mtime[8],
E_filesize[8],
E_maj[8],
E_min[8],
E_rmaj[8],
E_rmin[8],
E_namesize[8],
E_chksum[8],
E_name[EXPNLEN];
}

/* Tar header structure and format */
#define TBLOCK 512 /* length of tar header and data blocks */
#define TNAMLEN 100 /* maximum length for tar file names */
#define TMODLEN 8 /* length of mode field */
#define TUIDLEN 8 /* length of uid field */
#define TGIDLEN 8 /* length of gid field */
#define TSIZLEN 12 /* length of size field */
#define TTIMLEN 12 /* length of modification time field */
#define TCRCLEN 8 /* length of header checksum field */
/* tar header definition */
union tblock {
  char dummy[TBLOCK];
  struct header {
    char t_name[TNAMLEN]; /* name of file */
    char t_mode[TMODLEN]; /* mode of file */
    char t_uid[TUIDLEN]; /* uid of file */
    char t_gid[TGIDLEN]; /* gid of file */
    char t_size[TSIZLEN]; /* size of file in bytes */
    char t_mtime[TTIMLEN]; /* modification time of file */
    char t_chksum[TCRCLEN]; /* checksum of header */
    char t_typeflag; /* flag to indicate type of file */
    char t_linkname[TNAMLEN]; /* file this file is linked with */
    char t_magic[6]; /* magic string always "ustar" */
    char t_version[2]; /* version strings always "00" */
    char t uname[32]; /* owner of file in ASCII */
    char t_gname[32]; /* group of file in ASCII */
    char t_devmajor[8]; /* major number for special files */
    char t_devminor[8]; /* minor number for special files */
    char t_prefix[155]; /* pathname prefix */
  } tbuf;
};
/* volcopy tape label format and structure */
#define VMAGLEN 8
#define VVOLLEN 6
#define VFILLEN 464
struct volcopy_label {
  char v_magic[VMAGLEN],
  v_volume[VVOLLEN],
  v reels,
  v reel;
  long v_time,
  v length,
  v dens,
  v reelblks, /* u370 added field */
  v blksize, /* u370 added field */
  v nblocks; /* u370 added field */
  char v fill[VFILLEN];
```
archives(4)

long v_offset; /* used with -e and -reel options */
int v_type;    /* does tape have nbloks field */
};
The asetenv file is located in /usr/aset, the default operating directory of the Automated Security Enhancement Tool (ASET). An alternative working directory can be specified by the administrators through the set -d command or the ASETDIR environment variable. See set(1M). asetenv contains definitions of environment variables for ASET.

There are 2 sections in this file. The first section is labeled User Configurable Parameters. It contains, as the label indicates, environment variables that the administrators can modify to customize ASET behavior to suit their specific needs. The second section is labeled ASET Internal Environment Variables and should not be changed. The configurable parameters are explained as follows:

**TASK**

This variable defines the list of tasks that set will execute the next time it runs. The available tasks are:

- **tune**: Tighten system files.
- **usrgrp**: Check user/group.
- **sysconf**: Check system configuration file.
- **env**: Check environment.
- **cklist**: Compare system files checklist.
- **eeprom**: Check eeprom(1M) parameters.
- **firewall**: Disable forwarding of IP packets.

**CKLISTPATH_LOW**

These variables define the list of directories to be used by set to create a checklist file at the low, medium, and high security levels, respectively. Attributes of all the files in the directories defined by these variables will be checked periodically and any changes will be reported by set. Checks performed on these directories are not recursive. set only checks directories explicitly listed in these variables and does not check subdirectories of them.

**YPCHECK**

This variable is a boolean parameter. It specifies whether set should extend checking (when applicable) on system tables to their NIS equivalents or not. The value **true** enables it while the value **false** disables it.

**UID_ALIASES**

This variable specifies an alias file for user ID sharing. Normally, set warns about multiple user accounts.
sharing the same user ID because it is not advisable for accountability reason. Exceptions can be created using an alias file. User ID sharing allowed by the alias file will not be reported by aset. See asetmasters(4) for the format of the alias file.

### PERIODIC_SCHEDULE

This variable specifies the schedule for periodic execution of ASET. It uses the format of crontab(1) entries. Briefly speaking, the variable is assigned a string of the following format:

```
minutes hours day-of-month month day-of-week
```

Setting this variable does not activate the periodic schedule of ASET. To execute ASET periodically, aset(1M) must be run with the `-p` option. See aset(1M). For example, if `PERIODIC_SCHEDULE` is set to the following, and aset(1M) was started with the `-p` option, `aset` will run at 12:00 midnight every day:

```
0 0 * * *
```

### EXAMPLES

#### EXAMPLE 1  Sample asetenv file showing the settings of the ASET configurable parameters

The following is a sample `asetenv` file, showing the settings of the ASET configurable parameters:

```
CKLISTPATH_LOW=/etc:
CKLISTPATH_MED=$CHECKLISTPATH_LOW:/usr/bin:/usr/ucb
CKLISTPATH_HIGH=$CHECKLISTPATH_MED:/usr/lib:/usr/sbin
YPCHECK=false
UID_ALIASES=/usr/aset/masters/uid_aliases
PERIODIC_SCHEDULE="0 0 * * *
TASKS="env sysconf usgrp"
```

When `aset -p` is run with this file, `aset` is executed at midnight of every day. The `/` and `/etc` directories are checked at the low security level; the `/`, `/etc`, `/usr/bin`, and `/usr/ucb` directories are checked at the medium security level; and the `/`, `/etc`, `/usr/bin`, `/usr/lib`, and `/usr/sbin` directories are checked at the high security level. Checking of NIS system files is disabled. The `/usr/aset/masters/uid_aliases` file specifies the used IDs available for sharing. The `env`, `sysconf`, and `usgrp` tasks will be performed, checking the environment variables, various system tables, and the local `passwd` and `group` files.

### SEE ALSO

crontab(1), aset(1M), asetmasters(4)

ASET Administrator Manual
The /usr/aset masters directory contains several files used by the Automated Security Enhancement Tool (ASET). /usr/aset is the default operating directory for ASET. An alternative working directory can be specified by the administrators through the aset -d command or the ASETDIR environment variable. See aset(1M).

These files are provided by default to meet the need of most environments. The administrators, however, can edit these files to meet their specific needs. The format and usage of these files are described below.

All the master files allow comments and blank lines to improve readability. Comment lines must start with a leading "#" character.

tune.low
tune.med
tune.high

These files are used by the tune task (see aset(1M)) to restrict the permission settings for system objects. Each file is used by ASET at the security level indicated by the suffix. Each entry in the files is of the form:

```
pathname mode owner group type
```

where

- **pathname** is the full pathname
- **mode** is the permission setting
- **owner** is the owner of the object
- **group** is the group of the object
- **type** is the type of the object. It can be symlink for a symbolic link, directory for a directory, or file for everything else.
Regular shell wildcard ("*", "?", ...) characters can be used in the pathname for multiple references. See sh(1). The mode is a five-digit number that represents the permission setting. Note that this setting represents a least restrictive value. If the current setting is already more restrictive than the specified value, ASET does not loosen the permission settings.

For example, if mode is 00777, the permission will not be changed, since it is always less restrictive than the current setting.

Names must be used for owner and group instead of numeric ID’s. ? can be used as a “don’t care” character in place of owner, group, and type to prevent ASET from changing the existing values of these parameters.

uid_alias

This file allows user ID’s to be shared by multiple user accounts. Normally, ASET discourages such sharing for accountability reason and reports user ID’s that are shared. The administrators can, however, define permissible sharing by adding entries to the file. Each entry is of the form:

uid=alias1=alias2=alias3= ...

where

uid is the shared user id
alias? is the user accounts sharing the user ID

For example, if sync and daemon share the user ID 1, the corresponding entry is:

1=sync=daemon

cklist.low
cklist.med
cklist.high

These files are used by the cklist task (see aset(1M)), and are created the first time the task is run at the low, medium, and high levels. When the cklist task is run, it compares the specified directory’s contents with the appropriate cklist .level file and reports any discrepancies.

EXAMPLES

EXAMPLE 1 Examples of Valid Entries for the tune .low, tune .med, and tune .high Files

The following is an example of valid entries for the tune .low, tune .med, and tune .high files:

/bin 00777 root staffsymlink
/etc 02755 root staffdirectory
/dev/sd* 00640 rootoperatorfile
EXAMPLE 1 Examples of Valid Entries for the tune.low, tune.med, and tune.high Files (Continued)

SEE ALSO aset(1M), asetenv(4)

ASET Administrator Manual
/etc/security/audit_class

is an ASCII system file that stores class definitions. Programs use the getauclassent(3BSM) routines to access this information.

The fields for each class entry are separated by colons. Each class entry is a bitmap and is separated from each other by a newline.

Each entry in the audit_class file has the form:

```
mask:name:descriptive
```

The fields are defined as follows:

- **mask**: The class mask.
- **name**: The class name.
- **description**: The description of the class.

The classes are now user-configurable. Each class is represented as a bit in the class mask which is an unsigned integer. Thus, there are 32 different classes available, plus two meta-classes -- all and no.

- **all**: represents a conjunction of all allowed classes, and is provided as a shorthand method of specifying all classes.
- **no**: is the “invalid” class, and any event mapped solely to this class will not be audited. (Turning auditing on to the all meta class will NOT cause events mapped solely to the no class to be written to the audit trail.)

**EXAMPLE 1** Sample of an audit_class file.

Here is a sample of an audit_class file:

```
0x00000000:mo:invalid class
0x00000001:fr:file read
0x00000002:fw:file write
0x00000004:fa:file attribute access
0x00000008:fm:file attribute modify
0x00000010:fc:file create
0x00000020:fd:file delete
0x00000040:cl:file close
0xffffffff:all:all classes
```

**FILES**
/etc/security/audit_class

**SEE ALSO**
bsmconv(1M), getauclassent(3BSM), audit_event(4)
It is possible to deliberately turn on the no class in the kernel, in which case the audit trail will be flooded with records for the audit event AUE_NULL.

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.
The `audit_control` file contains audit control information used by `auditd(1M)`. Each line consists of a title and a string, separated by a colon. There are no restrictions on the order of lines in the file, although some lines must appear only once. A line beginning with `#` is a comment.

Directory definition lines list the directories to be used when creating audit files, in the order in which they are to be used. The format of a directory line is:

```
dir:directory-name
```

`directory-name` is where the audit files will be created. Any valid writable directory can be specified.

The following configuration is recommended:

```
/etc/security/audit/server/files
```

where `server` is the name of a central machine, since audit files belonging to different servers are usually stored in separate subdirectories of a single audit directory. The naming convention normally has `server` be a directory on a server machine, and all clients mount `/etc/security/audit/server` at the same location in their local file systems. If the same server exports several different file systems for auditing, their `server` names will, of course, be different.

There are several other ways for audit data to be arranged: some sites may have needs more in line with storing each host's audit data in separate subdirectories. The audit structure used will depend on each individual site.

The audit threshold line specifies the percentage of free space that must be present in the file system containing the current audit file. The format of the threshold line is:

```
minfree:percentage
```

where `percentage` is indicates the amount of free space required. If free space falls below this threshold, the audit daemon `auditd(1M)` invokes the shell script `audit_warn(1M)`. If no threshold is specified, the default is 0%.

The audit flags line specifies the default system audit value. This value is combined with the user audit value read from `audit_user(4)` to form the process audit state. The user audit value overrides the system audit value. The format of a flags line is:

```
flags:audit-flags
```

where `audit-flags` specifies which event classes are to be audited. The character string representation of `audit-flags` contains a series of flag names, each one identifying a single audit class, separated by commas. A name preceded by `−` means that the class should be audited for failure only; successful attempts are not audited. A name
preceded by `+' means that the class should be audited for success only; failing attempts are not audited. Without a prefix, the name indicates that the class is to be audited for both successes and failures. The special string `all' indicates that all events should be audited; `−all' indicates that all failed attempts are to be audited, and `+all' all successful attempts. The prefixes `^', `^−', and `^+' turn off flags specified earlier in the string (`^' and `^+' for failing and successful attempts, `^' for both). They are typically used to reset flags.

The non-attributable flags line is similar to the flags line, but this one contain the audit flags that define what classes of events are audited when an action cannot be attributed to a specific user. The format of a naflags line is:

naflags: audit-flags

The flags are separated by commas, with no spaces.

The following table lists the predefined audit classes:

<table>
<thead>
<tr>
<th>short name</th>
<th>long name</th>
<th>short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>no_class</td>
<td>null value for turning off event preselection</td>
</tr>
<tr>
<td>fr</td>
<td>file_read</td>
<td>Read of data, open for reading, etc.</td>
</tr>
<tr>
<td>fw</td>
<td>file_write</td>
<td>Write of data, open for writing, etc.</td>
</tr>
<tr>
<td>fa</td>
<td>file_attr_acc</td>
<td>Access of object attributes: stat, pathconf, etc.</td>
</tr>
<tr>
<td>fm</td>
<td>file_attr_mod</td>
<td>Change of object attributes: chown, flock, etc.</td>
</tr>
<tr>
<td>fc</td>
<td>file_creation</td>
<td>Creation of object</td>
</tr>
<tr>
<td>fd</td>
<td>file_deletion</td>
<td>Deletion of object</td>
</tr>
<tr>
<td>cl</td>
<td>file_close</td>
<td>close(2) system call</td>
</tr>
<tr>
<td>pc</td>
<td>process</td>
<td>Process operations: fork, exec, exit, etc.</td>
</tr>
<tr>
<td>nt</td>
<td>network</td>
<td>Network events: bind, connect, accept, etc.</td>
</tr>
<tr>
<td>ip</td>
<td>ipc</td>
<td>System V IPC operations</td>
</tr>
<tr>
<td>na</td>
<td>non_attrib</td>
<td>non-attributable events</td>
</tr>
<tr>
<td>ad</td>
<td>administrative</td>
<td>administrative actions: mount, exportfs, etc.</td>
</tr>
<tr>
<td>lo</td>
<td>login_logout</td>
<td>Login and logout events</td>
</tr>
<tr>
<td>ap</td>
<td>application</td>
<td>Application auditing</td>
</tr>
<tr>
<td>io</td>
<td>ioctl</td>
<td>ioctl(2) system call</td>
</tr>
<tr>
<td>ex</td>
<td>exec</td>
<td>exec(2) system call</td>
</tr>
<tr>
<td>ot</td>
<td>other</td>
<td>Everything else</td>
</tr>
<tr>
<td>all</td>
<td>all</td>
<td>All flags set</td>
</tr>
</tbody>
</table>

Note that the classes are configurable, see audit_class(4).

EXAMPLE 1 Sample /etc/security/audit_control File For the Machine eggplant

Here is a sample /etc/security/audit_control file for the machine eggplant:

dir: /etc/security/jedgar/eggplant
dir: /etc/security/jedgar.aux/eggplant
# # Last-ditch audit file system when jedgar fills up.
#
dir: /etc/security/global/eggplant
minfree: 20
flags: lo,ad,−all,`^−fm
naflags: lo,ad
EXAMPLE 1  Sample /etc/security/audit_control File For the Machine eggplant  
(Continued)

This identifies server jedgar with two file systems normally used for audit data, 
another server global used only when jedgar fills up or breaks, and specifies that 
the warning script is run when the file systems are 80% filled. It also specifies that all 
logins, administrative operations are to be audited (whether or not they succeed), and 
that failures of all types except failures to access object attributes are to be audited.

FILES  
/etc/security/audit_control  
/etc/security/audit_warn  
/etc/security/audit/*/*/  
/etc/security/audit_user  

SEE ALSO  
audit(1M), audit_warn(1M), auditd(1M), bsmconv(1M), audit(2), 
getauditflags(3BSM), audit.log(4), audit_class(4), audit_user(4)  

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.
The `audit_data` file contains information about the audit daemon. The file contains the process ID of the audit daemon, and the pathname of the current audit log file. The format of the file is:

```
<pid>:<pathname>
```

Where `pid` is the process ID for the audit daemon, and `pathname` is the full pathname for the current audit log file.

**EXAMPLE 1**

A sample `audit_data` file.

```
64:/etc/security/audit/server1/19930506081249.19930506230945.bongos
```

**FILES**

```
/proc/audit_data
```

**SEE ALSO**

`audit(1M)`, `auditd(1M)`, `bsmconv(1M)`, `audit(2)`, `audit.log(4)`

**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.
audit_event(4)

NAME
audit_event – audit event definition and class mapping

SYNOPSIS
/etc/security/audit_event

DESCRIPTION
/etc/security/audit_event is an ASCII system file that stores event definitions and specifies the event to class mappings. Programs use the getauevent(3BSM) routines to access this information.

The fields for each event entry are separated by colons. Each event is separated from the next by a newline.

Each entry in the audit_event file has the form:

number:name:description:flags

The fields are defined as follows:

number The event number.
name The event name.
description The description of the event.
flags Flags specifying classes to which the event is mapped.

EXAMPLES
EXAMPLE 1 Sample of the audit_event file entries.

Here is a sample of the audit_event file entries:

7:AUE_EXEC:exec(2):pc,ex
79:AUE_OPEN_WTC:open(2) - write, creat, trunc: fc, fd, fw
6152:AUE_login:login - success or failure: lo
6153:AUE_logout:logout: lo
6154:AUE_telnet:login - through telnet: lo
6155:AUE_rlogin:login - through rlogin: lo

FILES
/etc/security/audit_event

SEE ALSO
bsmconv(1M), getauevent(3BSM), audit_control(4)

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.
audit.log file are the depository for audit records stored locally or on an audit server. These files are kept in directories named in the file audit_control(4). They are named to reflect the time they are created and are, when possible, renamed to reflect the time they are closed as well. The name takes the form

```
yyyyymmddhhmmss.not_terminated.hostname
```

when open or if the auditd(1M) terminated ungracefully, and the form

```
yyyyymmddhhmmss.yyyyyymmddhhmmss.hostname
```

when properly closed. yyyy is the year, mm the month, dd day in the month, hh hour in the day, mm minute in the hour, and ss second in the minute. All fields are of fixed width.

The audit.log file begins with a standalone file token and typically ends with one also. The beginning file token records the pathname of the previous audit file, while the ending file token records the pathname of the next audit file. If the file name is NULL the appropriate path was unavailable.

The audit.log files contains audit records. Each audit record is made up of audit tokens. Each record contains a header token followed by various data tokens. Depending on the audit policy in place by auditon(2), optional other tokens such as trailers or sequences may be included.

The tokens are defined as follows:

The file token consists of:

- **token ID**: 1 byte
- **seconds of time**: 4 bytes
- **milliseconds of time**: 4 bytes
- **file name length**: 2 bytes
- **file pathname**: N bytes + 1 terminating NULL byte

The header token consists of:

- **token ID**: 1 byte
- **record byte count**: 4 bytes
- **version #**: 1 byte [2]
- **event type**: 2 bytes
- **event modifier**: 2 bytes
- **seconds of time**: 4 bytes (32-bit/64-bit value)
- **milliseconds of time**: 4 bytes (32-bit/64-bit value)

The expanded header token consists of:
The trailer token consists of:

- **token ID**: 1 byte
- **trailer magic number**: 2 bytes
- **record byte count**: 4 bytes

The arbitrary data token is defined:

- **token ID**: 1 byte
- **how to print**: 1 byte
- **basic unit**: 1 byte
- **unit count**: 1 byte
- **data items**: (depends on basic unit)

The in_addr token consists of:

- **token ID**: 1 byte
- **internet address**: 4 bytes

The expanded in_addr token consists of:

- **token ID**: 1 byte
- **IP address type/length**: 4 bytes
- **IP address**: 16 bytes

The ip token consists of:

- **token ID**: 1 byte
- **version and ihl**: 1 byte
- **type of service**: 1 byte
- **length**: 2 bytes
- **id**: 2 bytes
- **offset**: 2 bytes
- **ttl**: 1 byte
- **protocol**: 1 byte
- **checksum**: 2 bytes
- **source address**: 4 bytes
- **destination address**: 4 bytes

The expanded ip token consists of:

- **token ID**: 1 byte
- **version and ihl**: 1 byte
- **type of service**: 1 byte
- **length**: 2 bytes
- **id**: 2 bytes
- **offset**: 2 bytes
- **ttl**: 1 byte
<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>protocol</td>
<td>1 byte</td>
</tr>
<tr>
<td>checksum</td>
<td>2 bytes</td>
</tr>
<tr>
<td>address type/type</td>
<td>4 bytes</td>
</tr>
<tr>
<td>source address</td>
<td>4 bytes/16 bytes</td>
</tr>
<tr>
<td>address type/length</td>
<td>4 bytes</td>
</tr>
<tr>
<td>destination address</td>
<td>4 bytes/16 bytes</td>
</tr>
</tbody>
</table>

The **iport** token consists of:

- token ID             | 1 byte
- port IP address      | 2 bytes

The **path** token consists of:

- token ID             | 1 byte
- path length          | 2 bytes
- path                 | N bytes + 1 terminating NULL byte

The **process** token consists of:

- token ID             | 1 byte
- audit ID             | 4 bytes
- effective user ID    | 4 bytes
- effective group ID   | 4 bytes
- real user ID         | 4 bytes
- real group ID        | 4 bytes
- process ID           | 4 bytes
- session ID           | 4 bytes
- terminal ID          | 4 bytes
  - port ID             | 4 bytes/8 bytes (32-bit/64-bit value)
  - machine address     | 4 bytes

The **expanded process** token consists of:

- token ID             | 1 byte
- audit ID             | 4 bytes
- effective user ID    | 4 bytes
- effective group ID   | 4 bytes
- real user ID         | 4 bytes
- real group ID        | 4 bytes
- process ID           | 4 bytes
- session ID           | 4 bytes
- terminal ID          | 4 bytes
  - port ID             | 4 bytes/8 bytes (32-bit/64-bit value)
  - address type/length | 4 bytes
  - machine address     | 16 bytes

The **return** token consists of:

- token ID             | 1 byte
- error number         | 1 byte
- return value         | 4 bytes/8 bytes (32-bit/64-bit value)

The **subject** token consists of:

- token ID             | 1 byte
- audit ID             | 4 bytes
- effective user ID    | 4 bytes
effective group ID 4 bytes
real user ID 4 bytes
real group ID 4 bytes
process ID 4 bytes
session ID 4 bytes
terminal ID
  port ID 4 bytes/8 bytes (32-bit/64-bit value)
  machine address 4 bytes

The expanded subject token consists of:
  token ID 1 byte
  audit ID 4 bytes
  effective user ID 4 bytes
  effective group ID 4 bytes
  real user ID 4 bytes
  real group ID 4 bytes
  process ID 4 bytes
  session ID 4 bytes
terminal ID
  port ID 4 bytes/8 bytes (32-bit/64-bit value)
  address type/length 4 bytes
  machine address 16 bytes

The System V IPC token consists of:
  token ID 1 byte
  object ID type 1 byte
  object ID 4 bytes

The text token consists of:
  token ID 1 byte
  text length 2 bytes
  text N bytes + 1 terminating NULL byte

The attribute token consists of:
  token ID 1 byte
  file access mode 4 bytes
  owner user ID 4 bytes
  owner group ID 4 bytes
  file system ID 4 bytes
  node ID 8 bytes
  device 4 bytes/8 bytes (32-bit/64-bit)

The groups token consists of:
  token ID 1 byte
  number groups 2 bytes
  group list N * 4 bytes

The System V IPC permission token consists of:
  token ID 1 byte
  owner user ID 4 bytes
  owner group ID 4 bytes
  creator user ID 4 bytes
creator group ID 4 bytes
access mode 4 bytes
slot sequence # 4 bytes
key 4 bytes

The **arg** token consists of:

- token ID 1 byte
- argument # 1 byte
- argument value 4 bytes/8 bytes (32-bit/64-bit value)
- text length 2 bytes
- text N bytes + 1 terminating NULL byte

The **exec_args** token consists of:

- token ID 1 byte
- count 4 bytes
- text count null-terminated string(s)

The **exec_env** token consists of:

- token ID 1 byte
- count 4 bytes
- text count null-terminated string(s)

The **exit** token consists of:

- token ID 1 byte
- status 4 bytes
- return value 4 bytes

The **socket** token consists of:

- token ID 1 byte
- socket type 2 bytes
- remote port 2 bytes
- remote Internet address 4 bytes

The expanded **socket** token consists of:

- token ID 1 byte
- socket type 2 bytes
- local port 2 bytes
- address type/length 4 bytes
- local Internet address 4 bytes/16 bytes (IPv4/IPv6 address)
- remote port 4 bytes
- address type/length 4 bytes
- remote Internet address 4 bytes/16 bytes (IPv4/IPv6 address)

The **seq** token consists of:

- token ID 1 byte
- sequence number 4 bytes

**SEE ALSO**

audit(1M), auditd(1M), bsmconv(1M), audit(2), auditon(2), au_to(3BSM),
audit_control(4)

**NOTES**
Each token is generally written using the au_to(3BSM) family of function calls.
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.
### NAME
audit_user – per-user auditing data file

### SYNOPSIS
/etc/security/audit_user

### DESCRIPTION
audit_user is an access-restricted database that stores per-user auditing preselection data. The audit_user file can be used with other authorization sources, including the NIS map audit_user.byname and the NIS+ table audit_user. Programs use the getauusernam(3BSM) routines to access this information.

The search order for multiple user audit information sources is specified in the /etc/nsswitch.conf file, as described in the nsswitch.conf(4) man page. The lookup follows the search order for passwd(4).

The fields for each user entry are separated by colons (:). Each user is separated from the next by a newline. audit_user does not have general read permission.

Each entry in the audit_user file has the form:

```
username:always-audit-flags:never-audit-flags
```

The fields are defined as follows:

- `username` The user’s login name.
- `always-audit-flags` Flags specifying event classes to always audit.
- `never-audit-flags` Flags specifying event classes to never audit.

For a complete description of the audit flags and how to combine them, see the audit_control(4) man page.

### EXAMPLES
**EXAMPLE 1** Sample audit_user file

```
other:lo,ad:io,cl
fred:lo,ex,+fc,-fr,-fa:io,cl
ethyl:lo,ex,nt:io,cl
```

### FILES
/etc/nsswitch.conf
/etc/passwd
/etc/security/audit_user

### SEE ALSO
bsmconv(1M), getauusernam(3BSM), audit_control(4), nsswitch.conf(4), passwd(4)

### 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.
auth_attr(4)

NAME auth_attr – authorization description database

SYNOPSIS /etc/security/auth_attr

DESCRIPTION /etc/security/auth_attr is a local source for authorization names and
descriptions. The auth_attr file can be used with other authorization sources,
including the auth_attr NIS map and NIS+ table. Programs use the
getauthattr(3SECDB) routines to access this information.

The search order for multiple authorization sources is specified in the
/etc/nsswitch.conf file, as described in the nsswitch.conf(4) man page.

An authorization is a right assigned to users that is checked by certain privileged
programs to determine whether users can execute restricted functionality. Each entry
in the auth_attr database consists of one line of text containing six fields separated
by colons (:). Line continuations using the backslash (\) character are permitted. The
format of each entry is:

name:res1:res2:short_desc:long_desc:attr

name The name of the authorization. Authorization names are unique
strings. Construct authorization names using the following
convention:

prefix or prefix.suffix

prefix Everything in the name field up to the final dot (.)
Authors from Sun Microsystems, Inc. use solaris as a prefix. To avoid name conflicts, all other
authorizations should use a prefix that begins with the reverse–order Internet domain name of the
organization that creates the authorization (for example, com.xyzcompany). Prefixes can have
additional arbitrary components chosen by the authorization’s developer, with components separated
by dots.

suffix The final component in the name field. Specifies what is being authorized.

When there is no suffix, the name is defined as a
heading. Headings are not assigned to users but are
constructed for use by applications in their GUIs.

When a name ends with the word grant, the entry defines a grant
authorization. Grant authorizations are used to support
fine-grained delegation. Users with appropriate grant
<table>
<thead>
<tr>
<th>short_desc</th>
<th>A short description or terse name for the authorization. This name should be suitable for displaying in user interfaces, such as in a scrolling list in a GUI.</th>
</tr>
</thead>
<tbody>
<tr>
<td>long_desc</td>
<td>A long description. This field can explain the precise purpose of the authorization, the applications in which it is used, and the type of user that would be interested in using it. The long description can be displayed in the help text of an application.</td>
</tr>
<tr>
<td>attr</td>
<td>An optional list of semicolon-separated (;) key-value pairs that describe the attributes of an authorization. Zero or more keys may be specified. The keyword help identifies a help file in HTML. Help files can be read by a web browser using the URL:</td>
</tr>
<tr>
<td></td>
<td>file:/usr/lib/help/auths/locale/C/index.html</td>
</tr>
</tbody>
</table>

**EXAMPLES**

**EXAMPLE 1** Constructing a name

In the following example, the name has a prefix (`solaris.`) followed by a suffix (printer):

`solaris.printer`

**EXAMPLE 2** Defining a heading

Because the name field ends with a dot, the following entry defines a heading:

`solaris.hostmgr.:::Computers & Networks:::help=HostMgrHeader.html`

**EXAMPLE 3** Assigning separate authorizations to set user attributes

In this example, a heading entry is followed by other associated authorization entries. The entries below the heading provide separate authorizations for setting user attributes. The *attr* field for each entry, including the heading entry, assigns a help file. The application that uses the *help* key requires the value to equal the name of a file ending in `.htm` or `.html`:

`solaris.usermgr.:::Users, Groups & Email Aliases:::help=UserMgrHeader.html`
`solaris.usermgr.pswd:::Change User Passwords:::help=UserMgrPswd.html`
`solaris.usermgr.write:::Add, Modify & Delete:::help=UserMgrWrite.html`

**EXAMPLE 4** Assigning a grant authorization

This example assigns to an administrator the following authorizations:
EXAMPLE 4 Assigning a grant authorization  (Continued)

solaris.printmgr.grant
solaris.printmgr.admin
solaris.printmgr.nobanner
solaris.login.enable

With the above authorizations, the administrator can assign to others the
grant authorization, solaris.printmgr.grant, and the wildcard authorization,
solaris.printmgr.*, the administrator can grant to others any of the printer
authorizations. See user_attr(4) for more information about how wildcards can be
used to assign multiple authorizations whose names begin with the same components.

EXAMPLE 5 Authorizing the ability to assign other authorizations

The following entry defines an authorization that grants the ability to assign any
authorization created with a solaris prefix, when the administrator also has either
the specific authorization being granted or a matching wildcard entry:

solaris.grant:::Grant All Rights::help=PriAdmin.html

EXAMPLE 6 Consulting the local authorization file ahead of the NIS table

With the following entry from /etc/nsswitch.conf, the local auth_attr file is
consulted before the NIS table:

auth_attr:files nisplus

FILES

/etc/nsswitch.conf
/etc/user_attr
/etc/security/auth_attr

NOTES

When deciding which authorization source to use (see DESCRIPTION), keep in mind
that NIS+ provides stronger authentication than NIS.

Because the list of legal keys is likely to expand, any code that parses this database
must be written to ignore unknown key-value pairs without error. When any new
keywords are created, the names should be prefixed with a unique string, such as the
company’s stock symbol, to avoid potential naming conflicts.

Each application has its own requirements for whether the help value must be a
relative pathname ending with a filename or the name of a file. The only known
requirement is for the name of a file.
The following characters are used in describing the database format and must be escaped with a backslash if used as data: colon (:) semicolon (;), equals (=), and backslash (\).

SEE ALSO
getauthattr(3SECDB), getexecattr(3SECDB), getprofattr(3SECDB),
getuserattr(3SECDB), exec_attr(4), nsswitch.conf(4), user_attr(4)

auth_attr(4)
The bootparams file contains a list of client entries that diskless clients use for booting. Diskless booting clients retrieve this information by issuing requests to a server running the rpc.bootparamd(1M) program. The bootparams file may be used in conjunction with or in place of other sources for the bootparams information. See nsswitch.conf(4).

For each client the file contains an entry with the client’s name and a list of boot parameter values for that client. Each entry should have the form:

```
clientname  identifier-specifier ... 
```

The first item of each entry is the host name of the diskless client. The asterisk ("*") character may be used as a “wildcard” in place of the client name in a single entry. That entry will apply to all clients for whom there is not an entry that specifically names them.

This is followed by one or more whitespace characters and a series of identifier-specifiers separated by whitespace characters.

Each identifier-specifier has the form:

```
identifier-server:pathname 
```

or

```
identifier-domain-name 
```

The first form is used for file-specific identifiers. A file-specific identifier is a key that is used by diskless clients to identify a file or filesystem. server is the name of the server that will provide the file or filesystem to the diskless client, and pathname is the path to the exported file or filesystem on the specified server. The equal sign ("=") and colon (":") characters are used in the indicated positions. There should not be any whitespace within an identifier-specifier.

Non-file-specific identifiers use the second form of identifier-specifier. One non-file-specific value for identifier is supported: the assignment of the client’s domain name. In this case, the value used for identifier is domain. domain-name must be the client’s domain name. The algorithm for determining a client’s domain name is to first check for a domain identifier in the client-specific entry and then in “wildcard” entry. If none is found, the server’s domain name is used.
An entry may be split across multiple lines of the file. The backslash (\) character should be used as the last character of a line to signify that the entry continues on the next line. The line may only be split in places where whitespace is allowed in the entry.

A variation of the first form (identifier=server:pathname) is used for the ns key which forces sysidtool(1M) to use a specific name service. By default, sysidtool uses NIS+ in preference to NIS if it can find a NIS+ server for the system's domain on the subnet. This key may be necessary if you are trying to set up a hands-off installation, or if the name server is on a different subnet, which is common with NIS+.

If this key is not used, sysidtool uses broadcast to attempt to bind to either a NIS+ or NIS server; if a name server is not on the local subnet, which is possible for NIS+, the bind will fail, automatic configuration of the name service will fail, and an interactive screen is displayed, prompting the user to specify the name service.

The ns entry has the form:

ns=[server] : [nameservice] [ (netmask) ]

where:

server the name of a server that will provide a name service to bind to
nameservice the name service (nis, nisplus, or none);
netmask a series of four numbers separated by periods that specifies which portion of an IP address is the network part, and which is the host part.

The ns keyword can be set in add_install_client or by Host Manager.

EXAMPLES

**EXAMPLE 1** Example Of An Entry In The bootparams File

Here is an example of an entry in the bootparams file:

```plaintext
client1 root=server1:/export/client1/root 
   swap=server1:/export/client1/swap 
   domain=bldg1.workco.com 
   root=server2:/export/client2/root ns=::nis 
   root=server2:/export/client2/root ns=watson: 
   root=server2:/export/client2/root 
   ns=mach:nisplus(255.255.255.0)
```

FILES

/etc/bootparams

SEE ALSO

rpc.bootparamd(1M), sysidtool(1M), nsswitch.conf(4)

IA only

rpld(1M)
Solaris diskless clients use the identifiers "root", "swap", and "dump" to look up the pathnames for the root filesystem, a swap area, and a dump area, respectively. These are the only identifiers meaningful for SPARC diskless booting clients.

For IA booting clients, the additional keyword identifiers "numbootfiles," "bootfile," and "bootaddr" are used (see rpld(1M)).
The table of contents file, .cdtoc, is an ASCII file that describes the contents of a CD-ROM or other software distribution media. It resides in the top-level directory of the file system on a slice of a CD-ROM. It is independent of file system format, that is, the file system on the slice can be either UFS or HSFS.

Each entry in the .cdtoc file is a line that establishes the value of a parameter in the following form:

```
PARAM=value
```

Blank lines and comments (lines preceded by a pound-sign, "#") are also allowed in the file. Parameters are grouped by product, with the beginning of a product defined by a line of the form:

```
PRODNAME=value
```

Each product is expected to consist of one or more software packages that are stored together in a subdirectory on the distribution media. There can be any number of products described within the file. There is no required order in which the parameters must be specified, except that the parameters must be grouped by product and the PRODNAME parameter must appear first in the list of parameters for each product specified. Each parameter is described below. All of the parameters are required for each product.

**PRODNAME**

The full name of the product. This must be unique within the .cdtoc file and is preferably unique across all possible products. This value may contain white space. The length of this value is limited to 256 ASCII characters; other restrictions may apply (see below).

**PRODVERS**

The version of the product. The value can contain any combination of letters, numbers, or other characters. This value may contain white space. The length of this value is limited to 256 ASCII characters; other restrictions may apply (see below).

**PRODDIR**

The name of the top-level directory containing the product. This name should be relative to the top-level directory of the distribution media, for example, Solaris_2.6/Product. The number of path components in the name is limited only by the system’s maximum path name length, which is 1024 ASCII characters. Any single component is limited to 256 ASCII characters. This value cannot contain white space.
The lengths of the values of PRODNAME and PRODVERS are further constrained by the fact that the initial install programs and swmtool(1M) concatenate these values to produce the full product name. swmtool(1M) concatenates the two values (inserting a space) to produce the name displayed in its software selection menu, for example, Solaris 2.6. For unbundled products the combined length of the values of PRODNAME and PRODVERS must not exceed 256 ASCII characters.

When you install OS services with Solstice Host Manager, directories for diskless clients and Autoclient systems are created by constructing names derived from a concatenation of the values of PRODNAME, PRODVERS, and client architecture, for example, /export/exec/Solaris_2.x_sparc.all/usr/platform. The length of the component containing the product name and version must not exceed 256 ASCII characters. Thus, for products corresponding to bundled OS releases (for example, Solaris 2.4), the values of PRODNAME and PRODVERS are effectively restricted to lengths much less than 256.

The initial install programs and swmtool(1M) use the value of the PRODDIR macro in the .cdtoc file to indicate where packages can be found.

EXAMPLE 1 Sample of .cdtoc file.

Here is a sample .cdtoc file:

```
PRODNAME=Online DiskSuite
PRODVERS=2.0
PRODDIR=Online_DiskSuite_2.0

PRODNAME=Online Backup
PRODVERS=2.0
PRODDIR=Online_Backup_2.0
```

This example corresponds to the following directory layout on a CD-ROM partition:

```
/./SUNWmddr.c
/./SUNWmddr.m
/./SUNWmddu
/Online_DiskSuite_2.0
```

The bundled release of Solaris 2.6 includes the following .cdtoc file:

```
PRODNAME=Solaris
PRODVERS=2.6
PRODDIR=Solaris_2.6/Product
```

This file corresponds to the following directory layout on slice 0 of the Solaris 2.6 product CD:
EXAMPLE 1 Sample of .cdtoc file. (Continued)

```
./cdtoc
/Solaris_2.6/Product
  ./SUNWaccr
  ./SUNWaccu
  ./SUNWadmap
  .
  .
  ./SUNWutool
```

SEE ALSO swmtool(1M), clustertoc(4), packagetoc(4), pkginfo(4)
clustertoc is an ASCII file that describes a hierarchical view of a software product. A .clustertoc file is required for the base OS product. The file resides in the top-level directory containing the product.

The hierarchy described by .clustertoc can be of arbitrary depth, although the initial system installation programs assume that it has three levels. The hierarchy is described bottom-up, with the packages described in .packagetoc at the lowest layer. The next layer is the cluster layer which collects packages into functional units. The highest layer is the meta-cluster layer which collects packages and clusters together into typical configurations.

The hierarchy exists to facilitate the selection or deselection of software for installation at varying levels of granularity. Interacting at the package level gives the finest level of control over what software is to be installed.

Each entry in the .clustertoc file is a line that establishes the value of a parameter in the following form:

```
PARAM=value
```

A line starting with a pound-sign, “#”, is considered a comment and is ignored.

Parameters are grouped by cluster or meta-cluster. The start of a cluster description is defined by a line of the form:

```
CLUSTER=value
```

The start of a meta-cluster description is defined by a line of the form:

```
METACLUSTER=value
```

There is no order implied or assumed for specifying the parameters for a (meta-)cluster with the exception of the CLUSTER or METACLUSTER parameter, which must appear first and the END parameter which must appear last.

Each parameter is described below. All of the parameters are mandatory.

**CLUSTER**

The cluster identifier (for example, SUNWCacc). The identifier specified must be unique within the package and cluster identifier namespace defined by a product’s .packagetoc and .clustertoc files. The identifiers used are subject to the same constraints as those for package identifiers. These constraints are (from pkginfo(4)):

“All characters in the abbreviation must be alphanumeric and the first may not be numeric. The abbreviation is limited to a maximum length of nine characters. install, new, and all are reserved abbreviations.”
A cluster must be described before another cluster or meta-cluster may refer to it.

**METACLUSTER**

The metacluster identifier (for example, `SUNWCprog`). The identifier specified must be unique within the package and cluster identifier namespace defined by a product’s `.packagetoc` and `.clustertoc` files. The identifiers used are subject to the same constraints as those for package identifiers. These constraints are (from `pkginfo(4)`):

“All characters in the abbreviation must be alphanumeric and the first may not be numeric. The abbreviation is limited to a maximum length of nine characters. `install`, `new`, and `all` are reserved abbreviations.”

Meta-clusters cannot contain references to other meta-clusters.

**NAME**

The full name of the (meta-)cluster. The length of the name string supplied may not exceed 256 characters.

**VENDOR**

The name of the (meta-)cluster’s vendor. The length of the vendor string supplied may not exceed 256 characters.

**VERSION**

The version of the (meta-)cluster. The length of the version string supplied may not exceed 256 characters.

**DESC**

An informative textual description of the (meta-)cluster’s contents. The length of the description supplied may not exceed 256 characters. The text should contain no newlines.

**SUNW_CSRMEMBER**

Indicates that the package or cluster is a part of the (meta-)cluster currently being described. The value specified is the identifier of the package or cluster. There may be an arbitrary number of `SUNW_CSRMEMBER` parameters per (meta-)cluster.

**SUNW_CSRMBRIFF**

Indicates that the package is to be included dynamically in the (meta-)cluster currently being described. The value of this parameter must follow the following format:

```
SUNW_CSRMBRIFF=<<test> <test arc>> <package>
```

This line will be converted into a `SUNW_CSRMEMBER` entry at media installation time if the test provided matches the platform on which the media is being installed. There may be zero or more `SUN_CSRMBRIFF` parameters per (meta-)cluster.

---

File Formats 61
where the <test> is either the built-in test of "platform" or a shell script which returns shell true (0) or shell false (1) depending on the tests being performed in the script. <value> is passed to the test as the first argument and can be used to create a script that tests for multiple hardware objects. Finally <package> is the package that will be included in the final .clustertoc file as a SUNW_CSRMEMBER. See parse_dynamic_clustertoc(1M) for more information about the scripts.

EXAMPLE 1 A Cluster Description

The following is an example of a cluster description in a .clustertoc file.

```
CLUSTER=SUNWCacc
NAME=System Accounting
DESC=System accounting utilities
VENDOR=Sun Microsystems, Inc.
VERSION=7.2
SUNW_CSRMEMBER=SUNWaccr
SUNW_CSRMEMBER=SUNWaccu
END
```

EXAMPLE 2 A Meta-cluster Description

The following is an example of a meta-cluster description in a .clustertoc file.

```
METACLUSTER=SUNWCreq
NAME=Core System Support
DESC=A pre-defined software configuration consisting of the minimum required software for a standalone, non-networked workstation.
VENDOR=Sun Microsystems, Inc.
VERSION=2.x
SUNW_CSRMEMBER=SUNWadmr
SUNW_CSRMEMBER=SUNWcar
SUNW_CSRMEMBER=SUNWCcs
SUNW_CSRMEMBER=SUNWCcg6
SUNW_CSRMEMBER=SUNWCdfb
SUNW_CSRMEMBER=SUNWkvm
SUNW_CSRMEMBER=SUNWcnis
SUNW_CSRMEMBER=SUNWowdv
SUNW_CSRMEMBER=SUNWter
END
```

EXAMPLE 3 A Meta-cluster Description With a Dynamic Cluster Entry

The following is an example of a meta-cluster description with a dynamic cluster entry as indicated by the use of the SUNW_CSRMBRIFF parameter entries.

```
METACLUSTER=SUNWProg
NAME=Developer System Support
DESC=A pre-defined software configuration consisting of the typical software used by software developers.
VENDOR=Sun Microsystems, Inc.
VERSION=2.5
SUNW_CSRMEMBER=SUNWCadm
SUNW_CSRMBRIFF=(smcc.dctoc tcx)SUNWctx
```
EXAMPLE 3 A Meta-cluster Description With a Dynamic Cluster Entry

SUNW_CSRMBRIFF=(smcc.dctoc leo)SUNWClleo
SUNW_CSRMBRIFF=(smcc.dctoc sx)SUNWCsx
...
END

SEE ALSO
parse_dynamic_clustertoc(1M), cdtoc(4), order(4), packagetoc(4),
pkginfo(4)

NOTES
The current implementation of the initial system installation programs depend on the .clustertoc describing three required meta-clusters for the base OS product:

*SUNWCall* Contains all of the software packages in the OS distribution.

*SUNWCuser* Contains the typical software packages for an end-user of the OS distribution.

*SUNWCreq* Contains the bare-minimum packages required to boot and configure the OS to the point of running a multi-user shell.
compver is an ASCII file used to specify previous versions of the associated package which are upward compatible. It is created by a package developer. Each line of the file specifies a previous version of the associated package with which the current version is backward compatible.

Since some packages may require installation of a specific version of another software package, compatibility information is extremely crucial. Consider, for example, a package called "A" which requires version "1.0" of application "B" as a prerequisite for installation. If the customer installing "A" has a newer version of "B" (version 1.3), the compver file for "B" must indicate that "1.3" is compatible with version "1.0" in order for the customer to install package "A".

EXAMPLE 1 Sample compver file.

A sample compver file is shown below:

```
Version 1.3
Version 1.0
```

SEE ALSO pkginfo(4)

Application Packaging Developer’s Guide

NOTES The comparison of the version string disregards white space and tabs. It is performed on a word-by-word basis. Thus, "Version 1.3" and "Version 1.3" would be considered the same.

The entries in the compver file must match the values assigned to the VERSION parameter in the pkginfo(4) files.
**NAME**  
copyright – copyright information file

**DESCRIPTION**  
copyright is an ASCII file used to provide a copyright notice for a package. The text may be in any format. The full file contents (including comment lines) are displayed on the terminal at the time of package installation.

**SEE ALSO**  
*Application Packaging Developer’s Guide*
The operating system writes out a core image of a process when it is terminated due to
the receipt of some signals. The core image is called core and is written in the
process’s working directory (provided it can be; normal access controls apply). A
process with an effective user ID different from the real user ID will not produce a core
image. This is also true for a process with an effective group ID different from the real
group ID. Set-user-ID and set-group-ID programs do not produce core images either
when they terminate, since this would cause a security loophole.

The core file contains all the process information pertinent to debugging: contents of
hardware registers, process status, and process data. The format of a core file is object
file specific.

For ELF executable programs (see a.out(4)), the core file generated is also an ELF file,
containing ELF program and file headers. The e_type field in the file header has type
ET_CORE. The program header contains an entry for every segment that was part of
the process address space, including shared library segments. The contents of the
writable segments are also part of the core image.

The program header of an ELF core file also contains entries for two NOTE segments,
each containing several note entries as described below. The note entry header and
core file note type (n_type) definitions are contained in <sys/elf.h>. The first
NOTE segment exists for binary compatibility with old programs that deal with core
files. It contains structures defined in <sys/old_procfs.h>. New programs should
recognize and skip this NOTE segment, advancing instead to the new NOTE segment.
The old NOTE segment will be deleted from core files in a future release.

The old NOTE segment contains the following entries. Each has entry name "CORE"
and presents the contents of a system structure:

<table>
<thead>
<tr>
<th>Type</th>
<th>Entry Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prpsinfo_t</td>
<td>n_type: NT_PRPSINFO</td>
<td>This entry contains information of interest to the ps(1) command, such as process status, CPU usage, “nice” value, controlling terminal, user-ID, process-ID, the name of the executable, and so forth. The prpsinfo_t structure is defined in &lt;sys/old_procfs.h&gt;.</td>
</tr>
<tr>
<td>char array</td>
<td>n_type: NT_PLATFORM</td>
<td>This entry contains a string describing the specific model of the hardware platform on which this core file was created. This information is the same as provided by sysinfo(2) when invoked with the command SI_PLATFORM.</td>
</tr>
<tr>
<td>auxv_t array</td>
<td>n_type: NT_AUXV</td>
<td>This entry contains the array of auxv_t structures that was passed by the operating system as startup information to the dynamic linker. Auxiliary vector information is defined in &lt;sys/auxv.h&gt;.</td>
</tr>
</tbody>
</table>
Following these entries, for each light-weight process (LWP) in the process, the old NOTE segment contains an entry with a prstatus_t structure, plus other optionally-present entries describing the LWP, as follows:

- **prstatus_t**
  - n_type: NT_PRSTATUS. This structure contains things of interest to a debugger from the operating system, such as the general registers, signal dispositions, state, reason for stopping, process-ID, and so forth. The prstatus_t structure is defined in `<sys/old_procfs.h>`.

- **prfpregset_t**
  - n_type: NT_PRFPREG. This entry is present only if the LWP used the floating-point hardware. It contains the floating-point registers. The prfpregset_t structure is defined in `<sys/procfs_isa.h>`.

- **gwindows_t**
  - n_type: NT_GWINDOWS. This entry is present only on a SPARC machine and only if the system was unable to flush all of the register windows to the stack. It contains all of the unspilled register windows. The gwindows_t structure is defined in `<sys/regset.h>`.

- **prxregset_t**
  - n_type: NT_PRXREG. This entry is present only if the machine has extra register state associated with it. It contains the extra register state. The prxregset_t structure is defined in `<sys/procfs_isa.h>`.

The new NOTE segment contains the following entries. Each has entry name “CORE” and presents the contents of a system structure:

- **psinfo_t**
  - n_type: NT_PSINFO. This structure contains information of interest to the ps(1) command, such as process status, CPU usage, "nice" value, controlling terminal, user-ID, process-ID, the name of the executable, and so forth. The psinfo_t structure is defined in `<sys/procfs.h>`.

- **pstatus_t**
  - n_type: NT_PSTATUS. This structure contains things of interest to a debugger from the operating system, such as pending signals, state, process-ID, and so forth. The pstatus_t structure is defined in `<sys/procfs.h>`.

- **char array**
  - n_type: NT_PLATFORM. This entry contains a string describing the specific model of the hardware platform on which this core file was created. This information is the same as provided by sysinfo(2) when invoked with the command SI_PLATFORM.
auxv_t array
n_type: NT_AUXV. This entry contains the array of auxv_t structures that was passed by the operating system as startup information to the dynamic linker. Auxiliary vector information is defined in <sys/auxv.h>.

struct utsname
n_type: NT_UTSNAME. This structure contains the system information that would have been returned to the process if it had performed a uname(2) system call prior to dumping core. The utsname structure is defined in <sys/utsname.h>.

prcred_t
n_type: NT_PRCRED. This structure contains the process credentials, including the real, saved, and effective user and group IDs. The prcred_t structure is defined in <sys/procfs.h>. Following the structure is an optional array of supplementary group IDs. The total number of supplementary group IDs is given by the pr_ngroups member of the prcred_t structure, and the structure includes space for one supplementary group. If pr_ngroups is greater than 1, there will be pr_ngroups - 1 gid_t items following the structure; otherwise, there will be no additional data.

Following these entries, for each LWP in the process, the new NOTE segment contains an entry with an lwpsinfo_t structure plus an entry with an lwpstatus_t structure, plus other optionally-present entries describing the LWP, as follows:

lwpsinfo_t
n_type: NT_LWPSINFO. This structure contains information of interest to the ps(1) command, such as LWP status, CPU usage, “nice” value, LWP-ID, and so forth. The lwpsinfo_t structure is defined in <sys/procfs.h>.

lwpstatus_t
n_type: NT_LWPSTATUS. This structure contains things of interest to a debugger from the operating system, such as the general registers, the floating point registers, state, reason for stopping, LWP-ID, and so forth. The lwpstatus_t structure is defined in <sys/procfs.h>.

gwindows_t
n_type: NT_GWINDOWS. This entry is present only on a SPARC machine and only if the system was unable to flush all of the register windows to the stack. It contains all of the unspilled register windows. The gwindows_t structure is defined in <sys/regset.h>.

prxregset_t
n_type: NT_PRXREG. This entry is present only if the machine has extra register state associated with it. It contains the extra register state. The prxregset_t structure is defined in <sys/procfs_isa.h>.
asrset_t n_type: NT ASRS. This entry is present only on a SPARC V9 machine and only if the process is a 64-bit process. It contains the ancillary state registers for the LWP. The asrset_t structure is defined in <sys/regset.h>.

The size of the core file created by a process may be controlled by the user (see getrlimit(2)).

SEE ALSO
adb(1), gcore(1), ps(1), crash(1M), getrlimit(2), setuid(2), sysinfo(2), uname(2), elf(3ELF), a.out(4), proc(4), signal(3HEAD)

ANSI C Programmer's Guide
NAME  

dacf.conf – device auto-configuration configuration file

SYNOPSIS  

/etc/dacf.conf

DESCRIPTION

The kernel uses the `dacf.conf` file to automatically configure hot plugged devices. Because the `dacf.conf` file contains important kernel state information, it should not be modified.

The format of the `/etc/dacf.conf` file is not public and might change in versions of the Solaris operating environment that are not compatible with Solaris 8.

ATTRIBUTES

See `attributes(5)` for 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>
</tbody>
</table>

SEE ALSO  

`attributes(5)`

NOTES

This document does not constitute an API. The `/etc/dacf.conf` file might not exist or might contain different contents or interpretations in versions of the Solaris operating environment that are not compatible with Solaris 8. The existence of this notice does not imply that any other documentation lacking this notice constitutes an API.
default_fs(4)

NAME  default_fs, fs – specify the default file system type for local or remote file systems

DESCRIPTION  When file system administration commands have both specific and generic components (for example, fsck(1M)), the file system type must be specified. If it is not explicitly specified using the -F FSTYPE command line option, the generic command looks in /etc/vfstab in order to determine the file system type, using the supplied raw or block device or mount point. If the file system type can not be determined by searching /etc/vfstab, the command will use the default file system type specified in either /etc/default/fs or /etc/dfs/fstypes, depending on whether the file system is local or remote.

The default local file system type is specified in /etc/default/fs by a line of the form LOCAL=fstype (for example, LOCAL=ufs). The default remote file system type is determined by the first entry in the /etc/dfs/fstypes file.

File system administration commands will determine whether the file system is local or remote by examining the specified device name. If the device name starts with "/'" (slash), it is considered to be local; otherwise it is remote.

The default file system types can be changed by editing the default files with a text editor.

FILES  /etc/vfstab  list of default parameters for each file system
        /etc/default/fs  the default local file system type
        /etc/dfs/fstypes  the default remote file system type

SEE ALSO  fsck(1M), fstypes(4), vfstab(4)
defaultrouter(4)

NAME  defaultrouter – configuration file for default router(s)

SYNOPSIS  /etc/defaultrouter

DESCRIPTION  The /etc/defaultrouter file defines the default routers the system will use.

The format of the file is as follows:

The /etc/defaultrouter file can contain the hostnames or IP addresses of one or more default routers, separated by white space. If you use hostnames, each hostname must also be listed in the local /etc/hosts file, because no name services are running at the time that this script is run.

Lines beginning with the “#” character are treated as comments.

The default routes listed in this file replace those added by the kernel during diskless booting. An empty /etc/defaultrouter file will cause the default route added by the kernel to be deleted.

FILES  /etc/defaultrouter
       Configuration file containing the hostnames or IP addresses of one or more default routers.

SEE ALSO  hosts(4)
depend is an ASCII file used to specify information concerning software dependencies for a particular package. The file is created by a software developer.

Each entry in the depend file describes a single software package. The instance of the package is described after the entry line by giving the package architecture and/or version. The format of each entry and subsequent instance definition is:

\[\text{type pkg name (arch)version (arch)version ...}\]

The fields are:

\textbf{type} \hspace{1cm} Defines the dependency type. Must be one of the following characters:

\begin{itemize}
  \item \textbf{P} Indicates a prerequisite for installation; for example, the referenced package or versions must be installed.
  \item \textbf{I} Implies that the existence of the indicated package or version is incompatible.
  \item \textbf{R} Indicates a reverse dependency. Instead of defining the package's own dependencies, this designates that another package depends on this one. This type should be used only when an old package does not have a depend file, but relies on the newer package nonetheless. Therefore, the present package should not be removed if the designated old package is still on the system since, if it is removed, the old package will no longer work.
\end{itemize}

\textbf{pkg} \hspace{1cm} Indicates the package abbreviation.

\textbf{name} \hspace{1cm} Specifies the full package name.

\textbf{(arch)version} \hspace{1cm} Specifies a particular instance of the software. A version name cannot begin with a left parenthesis. The instance specifications, both (arch) and version, are completely optional, but each (arch)version pair must begin on a new line that begins with white space. A null version set equates to any version of the indicated package.

**EXAMPLE 1** Sample of depend file.

Here is a sample depend file:

\begin{verbatim}
#ident "@(#)pkg.compat:depend 1.1"
P nsu Networking Support Utilities
P inet Internet Utilities
P sys System Header Files
\end{verbatim}
EXAMPLE 1  Sample of depend file.  (Continued)

P  src_compat  Source Compatibility Files

SEE ALSO  Application Packaging Developer's Guide
The `device_allocate` file contains mandatory access control information about each physical device. Each device is represented by a one line entry of the form:

```
device-name;device-type;reserved;reserved;auths;device-exec
```

where

- **device-name**: This is an arbitrary ASCII string naming the physical device. This field contains no embedded white space or non-printable characters.
- **device-type**: This is an arbitrary ASCII string naming the generic device type. This field identifies and groups together devices of like type. This field contains no embedded white space or non-printable characters.
- **reserved**: This field is reserved for future use.
- **reserved**: This field is reserved for future use.
- **auths**: This field contains a comma-separated list of authorizations required to allocate the device, or an `*` symbol to indicate that the device is not allocatable, or an `@` symbol to indicate that no explicit authorization is needed to allocate the device.

The default authorization is `solaris.device.allocate`. See `auths(1)`.

- **device-exec**: This is the physical device’s data purge program to be run any time the device is acted on by `allocate(1M)`. This is to ensure that all usable data is purged from the physical device before it is reused. This field contains the filename of a program in `/etc/security/lib` or the full pathname of a cleanup script provided by the system administrator.

The `device_allocate` file is an ASCII file that resides in the `/etc/security` directory.

Lines in `device_allocate` can end with a `\` to continue an entry on the next line.

Comments may also be included. A `#` makes a comment of all further text until the next NEWLINE not immediately preceded by a `\`.

White space is allowed in any field.
device_allocate(4)

The device_allocate file must be created by the system administrator before device allocation is enabled.

The device_allocate file is owned by root, with a group of sys, and a mode of 0644.

EXAMPLES

EXAMPLE 1 Declaring an allocatable device

Declare that physical device st0 is a type st. st is allocatable, and the script used to clean the device after running deallocate(1M) is named /etc/security/lib/st_clean.

# scsi tape
st0;
  st;
    reserved;
    reserved;
    solaris.device.allocate;
    /etc/security/lib/st_clean;

EXAMPLE 2 Declaring an allocatable device with authorizations

Declare that physical device fd0 is of type fd. fd is allocatable by users with the solaris.device.allocate authorization, and the script used to clean the device after running deallocate(1M) is named /etc/security/lib/fd_clean.

# floppy drive
fd0;
  fd;
    reserved;
    reserved;
    &;
    /etc/security/lib/fd_clean;

Notice that making a device allocatable means that you need to allocate and deallocate it to use it (with allocate(1M) and deallocate(1M)). If a device is not allocatable, there will be an asterisk (*) in the auths field, and no one can use the device.

FILES

/etc/security/device_allocate  Contains list of allocatable devices

SEE ALSO

auths(1), allocate(1M), bsmconv(1M), deallocate(1M), list_devices(1M), auth_attr(4)

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.
device.cfinfo files pass information about device configuration to the devconfig(1M) program. They allow devconfig(1M) to provide the user with valid ranges for device attributes.

devconfig(1M) associates a device with its cfinfo file by name. For example, the device logi for the Logitec Bus Mouse has the devconfig(1M) configuration file logi.cfinfo associated with it in the DEVCONFIGHOME directory. DEVCONFIGHOME is /usr/lib/devconfig by default and may be set in the user’s environment.

Below is a yaccish grammar of a cfinfo file:

```
cfinfo_file:                 cfinfo_devspec EOF
                 ;

    cfinfo_devspec:       cfinfo_spec_list SEMICOLON
                 ;

    cfinfo_spec_list:     cfinfo_spec | 
                 cfinfo_spec_list cfinfo_spec
                 ;

    cfinfo_spec:          comment | 
                 attr_value_pair NEWLINE
                 ;

    comment:              POUNDSIGN | 
                 POUNDSIGN STRING
                 ;

    attr_value_pair:      ATTR_NAME_EQUALS_STRING | 
                 ATTR_OWNAME_EQUALS_STRING
                 ATTR_TITLE_EQUALS_STRING |
```

File Formats 77
ATTR_CATEGORY EQUALS STRING | 
ATTR_INSTANCE EQUALS STRING | 
ATTR_CLASS EQUALS STRING | 
ATTR_TYPE EQUALS STRING | 
ATTR_REAL EQUALS STRING | 
ATTR_AUTO EQUALS STRING | 
NAME EQUALS value_spec_string
;

value_spec_string: QUOTE value_spec QUOTE
;

value_spec: value_type COMMA value_list
;

value_type: | /* EMPTY */
 | TYPE_NUMERIC |
 | TYPE_STRING |
 | TYPE_VAR |
;

value_list: integer_value_list |
 | string_value_list |
;

integer_value_list: INTEGER |
 | INTEGER COLON INTEGER |
 | INTEGER COMMA integer_value_list |
;

device.c

info(4)

man pages section 4: File Formats • Last Revised 31 Dec 1996
string_value_list: STRING COMMA string_value_list

<table>
<thead>
<tr>
<th>ATTR_NAME</th>
<th>name</th>
<th># device name specified in driver.conf</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTR_CLASS</td>
<td>class</td>
<td># device class specified in driver.conf</td>
</tr>
<tr>
<td>ATTR_TYPE</td>
<td>type</td>
<td># device type specified in OWconfig</td>
</tr>
<tr>
<td>ATTR_OWNAME</td>
<td><strong>owname</strong></td>
<td># device name specified in OWconfig</td>
</tr>
<tr>
<td>ATTR_TITLE</td>
<td><strong>title</strong></td>
<td># device title displayed by devconfig</td>
</tr>
<tr>
<td>ATTR_CATEGORY</td>
<td><strong>category</strong></td>
<td># device category</td>
</tr>
<tr>
<td>ATTR_INSTANCE</td>
<td><strong>instance</strong></td>
<td># device unit</td>
</tr>
<tr>
<td>ATTR_REAL</td>
<td><strong>real</strong></td>
<td># attributes to write to driver.conf</td>
</tr>
<tr>
<td>ATTR_AUTO</td>
<td><strong>auto</strong></td>
<td># self-identifying device attribute</td>
</tr>
<tr>
<td>TYPE_NUMERIC</td>
<td>numeric</td>
<td># precedes an integer value list</td>
</tr>
<tr>
<td>TYPE_STRING</td>
<td>string</td>
<td># precedes a string values list</td>
</tr>
<tr>
<td>TYPE_VAR</td>
<td>var</td>
<td># precedes a variable specification</td>
</tr>
</tbody>
</table>

The first value in a value_list is the default value picked by devconfig(1M) for the attribute. An attribute name of the form __name__ is used internally by devconfig(1M). Number ranges are specified as n1:n2. An internal attribute of the type var specifies a configurable portion of a real attribute. (See examples below.) Certain internal attributes have an expanded form when displayed. These attributes are listed in the file abbreviations in DEVCONFIGHOME. The file abbreviations also includes a list of name mappings for certain category names. If the __real__ attribute is present, only the attribute names it specifies are written to a driver.conf file. Otherwise, all non-internal attributes are written.

**EXAMPLES**

**EXAMPLE 1** Device configuration file logi.cfinfo for the LOGITECH bus mouse.

Here is the device configuration file logi.cfinfo for the LOGITECH bus mouse. The driver configuration file for this device is called logi.conf.
EXAMPLE 1 Device configuration file logi.cfinfo for the LOGITECH bus mouse.

(Continued)

    name="logi"
    __owname__="pointer:0"
    __title__="Logitec bus mouse"
    __category__="pointer"
    class="sysbus"
    type="LOGI-B"
    buttons="var, __nbuttons__"
    __nbuttons__="numeric,2:3"
    dev="/dev/logi"
    intr="numeric,1", "var, __irq__"
    __irq__="numeric,2:5"
    __real__="name", "class", "intr"

The driver name for the LOGITECH Bus Mouse is logi. The device name in OWconfig (see the OpenWindows Desktop Reference Manual) is pointer: 0. The device category is pointer; the device category is displayed as pointing devices, however, since there is a category mapping for pointer in the abbreviations file. The device class is sysbus as specified in the file /kernel/drv/classes. A device of class own does not have a device driver associated with it. The device IPL is 1. The device IRQ is substituted by the variable __irq__ and has a range of 2 to 5. A name mapping for __irq__ exists in abbreviations and so __irq__ is displayed as Interrupt (IRQ). The device attributes written to logi.conf are name, class, and intr as specified by the __real__ entry.

The resulting entry in logi.conf is:

    name="logi" class="sysbus" intr=1,2;

The resulting entry in OWconfig is:

    type="LOGI-B" buttons=3 dev="/dev/logi" class="owin"
    name="pointer:0";

Here is an example of a self-identifying device.

    name="lp"
    __title__="Parallel printer port"
    __category__="lp"
    __class__="sysbus"
    __auto__="string,true"

The driver for the parallel port automatically identifies it, and devconfig(1M) treats this device as self-identifying.

FILES

abbreviations
ATTRIBUTES

See 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 devconfig(1M), driver.conf(4), attributes(5) OpenWindows Desktop Reference Manual

device.cfinfo(4)
device_maps(4)

NAME      device_maps – device_maps file
SYNOPSIS  /etc/security/device_maps
DESCRIPTION The device_maps file contains access control information about each physical device. Each device is represented by a one line entry of the form:

device-name : device-type : device-list :

where

device-name  This is an arbitrary ASCII string naming the physical device. This field contains no embedded white space or non-printable characters.
device-type This is an arbitrary ASCII string naming the generic device type. This field identifies and groups together devices of like type. This field contains no embedded white space or non-printable characters.
device-list  This is a list of the device special files associated with the physical device. This field contains valid device special file path names separated by white space.

The device_maps file is an ASCII file that resides in the /etc/security directory. Lines in device_maps can end with a ‘\’ to continue an entry on the next line.

Comments may also be included. A ‘#’ makes a comment of all further text until the next NEWLINE not immediately preceded by a ‘\’.

Leading and trailing blanks are allowed in any of the fields.

The device_maps file must be created by the system administrator before device allocation is enabled.

This file is owned by root, with a group of sys, and a mode of 0644.

EXAMPLES  EXAMPLE 1 A sample device_maps file

# scsi tape
st1:\
rmt:\
/dev/rst21 /dev/rst21 /dev/rst5 /dev/rst5 /dev/rst13 \
/dev/nrst21 /dev/nrst29 /dev/nrst29 /dev/nrst1l /dev/nrst1m \
/dev/rmt/1 /dev/rmt/1h /dev/rmt/1u /dev/rmt/1ln /dev/rmt/1mn \
/dev/rmt/1l /dev/rmt/1hn /dev/rmt/1un /dev/rmt/1l/ /dev/rmt/1ln:

FILES     /etc/security/device_maps
SEE ALSO  allocate(1M), bsmconv(1M), deallocate(1M), dminfo(1M), list_devices(1M)
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.
<table>
<thead>
<tr>
<th>NAME</th>
<th>dfstab – file containing commands for sharing resources across a network</th>
</tr>
</thead>
</table>
| DESCRIPTION | dfstab resides in directory /etc/dfs and contains commands for sharing resources across a network. dfstab gives a system administrator a uniform method of controlling the automatic sharing of local resources.

Each line of the dfstab file consists of a share(1M) command. The dfstab file can be read by the shell to share all resources. System administrators can also prepare their own shell scripts to execute particular lines from dfstab.

The contents of dfstab are executed automatically when the system enters run-level 3.

| SEE ALSO  | share(1M), shareall(1M) |
The /etc/dhcp/inittab file contains information about the Dynamic Host Configuration Protocol (DHCP) options, which are network configuration parameters passed from DHCP servers to DHCP clients when a client machine uses DHCP. Since many DHCP-related commands must parse and understand these DHCP options, this file serves as a central location where information about these options may be obtained.

The DHCP inittab file provides three general pieces of information:

- A mnemonic alias, or symbol name, for each option number. For instance, option 12 is aliased to the name Hostname. This is useful for DHCP-related programs that require human interaction, such as dhcpinfo(1).
- Information about the syntax for each option. This includes information such as the type of the value, for example, whether it is a 16-bit integer or an IP address.
- The policy for what options are visible to which DHCP-related programs.

The dhcp_inittab file can only be changed upon system upgrade. Only additions of SITE options (or changes to same) will be preserved during upgrade.

The VENDOR options defined here are intended for use by the Solaris DHCP client and DHCP management tools. The SUNW vendor space is owned by Sun, and changes are likely during upgrade. If you need to configure the Solaris DHCP server to support the vendor options of a different client, see dhctab(4) for details.

Each DHCP option belongs to a certain category, which roughly defines the scope of the option; for instance, an option may only be understood by certain hosts within a given site, or it may be globally understood by all DHCP clients and servers. The following categories are defined; the category names are not case-sensitive:

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STANDARD</td>
<td>All client and server DHCP implementations agree on the semantics. These are administered by the Internet Assigned Numbers Authority (IANA). These options are numbered from 1 to 127.</td>
</tr>
<tr>
<td>SITE</td>
<td>Within a specific site, all client and server implementations agree on the semantics. However, at another site the type and meaning of the option may be quite different. These options are numbered from 128 to 254.</td>
</tr>
<tr>
<td>VENDOR</td>
<td>Each vendor may define 254 options unique to that vendor. The vendor is identified within a DHCP packet by the &quot;Vendor Class&quot; option, number 60. An option with a specific numeric identifier belonging to one vendor will, in general, have a type and semantics different from that of a different vendor. Vendor options are &quot;super-encapsulated&quot; into the vendor field number 43, as</td>
</tr>
</tbody>
</table>
defined in RFC 2132. The dhcp_inittab file only contains Sun vendor options. Define non-Sun vendor options in the dhcptab file.

FIELD
This category allows the fixed fields within a DHCP packet to be aliased to a mnemonic name for use with dhcpinfo(1).

INTERNAL
This category is internal to the Solaris DHCP implementation and will not be further defined.

Data entries are written one per line and have seven fields; each entry provides information for one option. Each field is separated by a comma, except for the first and second, which are separated by whitespace (as defined in isspace(3C)). An entry cannot be continued onto another line. Blank lines and those whose first non-whitespace character is '#' are ignored.

The fields, in order, are:

- Mnemonic Identifier
  The Mnemonic Identifier is a user-friendly alias for the option number; it is not case sensitive. This field must be per-category unique and should be unique across all categories. The option names in the STANDARD, SITE, and VENDOR spaces should not overlap, or the behavior will be undefined. See Mnemonic Identifiers for Options section of this man page for descriptions of the option names.

- Category (scope)
  The Category field is one of STANDARD, SITE, VENDOR, FIELD, or INTERNAL and identifies the scope in which the option falls.

- Option Number
  The Option Number is the number of this option when it is in a DHCP packet. This field should be per-category unique and the STANDARD and SITE fields should not have overlapping code fields or the behavior is undefined.

- Data Type
  Data Type is one of the following values, which are not case sensitive:
  - Ascii: A printable character string
  - Octet: An array of bytes
  - Unumber8: An 8-bit unsigned integer
  - Snumber8: An 8-bit signed integer
  - Unumber16: A 16-bit unsigned integer
  - Snumber16: A 16-bit signed integer
  - Unumber32: A 32-bit unsigned integer
  - Snumber32: A 32-bit signed integer
Unumber64  A 64-bit unsigned integer
Snumber64  A 64-bit signed integer

Ip
An IP address The data type field describes an indivisible unit of the option payload, using one of the values listed above.

■ Granularity
The Granularity field describes how many "indivisible units" in the option payload make up a whole value or item for this option.

■ Maximum Number Of Items

■ Visibility
The Visibility field specifies which DHCP-related programs make use of this information, and should always be defined as "sdmi" for newly added options.

The following table maps the mnemonic identifiers used in Solaris DHCP to RFC-2132 options:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subnet</td>
<td>1</td>
<td>Subnet Mask, dotted Internet address (IP).</td>
</tr>
<tr>
<td>UTCoffst</td>
<td>2</td>
<td>Coordinated Universal time offset (seconds).</td>
</tr>
<tr>
<td>Router</td>
<td>3</td>
<td>List of Routers, IP.</td>
</tr>
<tr>
<td>Timeserv</td>
<td>4</td>
<td>List of RFC-868 servers, IP.</td>
</tr>
<tr>
<td>IEN116ns</td>
<td>5</td>
<td>List of IEN 116 name servers, IP.</td>
</tr>
<tr>
<td>DNSServ</td>
<td>6</td>
<td>List of DNS name servers, IP.</td>
</tr>
<tr>
<td>Logserv</td>
<td>7</td>
<td>List of MIT-LCS UDP log servers, IP.</td>
</tr>
<tr>
<td>Cookie</td>
<td>8</td>
<td>List of RFC-865 cookie servers, IP.</td>
</tr>
<tr>
<td>Lprserv</td>
<td>9</td>
<td>List of RFC-1179 line printer servers, IP.</td>
</tr>
<tr>
<td>Impress</td>
<td>10</td>
<td>List of Imagen Impress servers, IP.</td>
</tr>
<tr>
<td>Resource</td>
<td>11</td>
<td>List of RFC-887 resource location servers, IP.</td>
</tr>
<tr>
<td>Hostname</td>
<td>12</td>
<td>Client’s hostname, value from hosts database.</td>
</tr>
<tr>
<td>Bootsize</td>
<td>13</td>
<td>Number of 512 octet blocks in boot image, NUMBER.</td>
</tr>
<tr>
<td>Dumpfile</td>
<td>14</td>
<td>Path where core image should be dumped, ASCII.</td>
</tr>
<tr>
<td>DNSdomain</td>
<td>15</td>
<td>DNS domain name, ASCII.</td>
</tr>
<tr>
<td>Swapserv</td>
<td>16</td>
<td>Client’s swap server, IP.</td>
</tr>
<tr>
<td>Rootpath</td>
<td>17</td>
<td>Client’s Root path, ASCII.</td>
</tr>
<tr>
<td>Symbol</td>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>ExtendP</td>
<td>18</td>
<td>Extensions path, ASCII.</td>
</tr>
<tr>
<td>IpFwdF</td>
<td>19</td>
<td>IP Forwarding Enable/Disable, NUMBER.</td>
</tr>
<tr>
<td>NLrouteF</td>
<td>20</td>
<td>Non-local Source Routing, NUMBER.</td>
</tr>
<tr>
<td>PFfilter</td>
<td>21</td>
<td>Policy Filter, IP.</td>
</tr>
<tr>
<td>MaxIpSiz</td>
<td>22</td>
<td>Maximum datagram Reassembly Size, NUMBER.</td>
</tr>
<tr>
<td>IpTTL</td>
<td>23</td>
<td>Default IP Time to Live, (1=&lt;x=&lt;255), NUMBER.</td>
</tr>
<tr>
<td>PathTO</td>
<td>24</td>
<td>RFC-1191 Path MTU Aging Timeout, NUMBER.</td>
</tr>
<tr>
<td>PathTbl</td>
<td>25</td>
<td>RFC-1191 Path MTU Plateau Table, NUMBER.</td>
</tr>
<tr>
<td>MTU</td>
<td>26</td>
<td>Interface MTU, $x=\geq68$, NUMBER.</td>
</tr>
<tr>
<td>SameMtuF</td>
<td>27</td>
<td>All Subnets are Local, NUMBER.</td>
</tr>
<tr>
<td>Broadcast</td>
<td>28</td>
<td>Broadcast Address, IP.</td>
</tr>
<tr>
<td>MaskDscF</td>
<td>29</td>
<td>Perform Mask Discovery, NUMBER.</td>
</tr>
<tr>
<td>MaskSupF</td>
<td>30</td>
<td>Mask Supplier, NUMBER.</td>
</tr>
<tr>
<td>RDiscvyF</td>
<td>31</td>
<td>Perform Router Discovery, NUMBER.</td>
</tr>
<tr>
<td>RSolictS</td>
<td>32</td>
<td>Router Solicitation Address, IP.</td>
</tr>
<tr>
<td>StaticRt</td>
<td>33</td>
<td>Static Route, Double IP (network router).</td>
</tr>
<tr>
<td>TrailerF</td>
<td>34</td>
<td>Trailer Encapsulation, NUMBER.</td>
</tr>
<tr>
<td>ArpTimeO</td>
<td>35</td>
<td>ARP Cache Time out, NUMBER.</td>
</tr>
<tr>
<td>EthEncap</td>
<td>36</td>
<td>Ethernet Encapsulation, NUMBER.</td>
</tr>
<tr>
<td>TcpTTL</td>
<td>37</td>
<td>TCP Default Time to Live, NUMBER.</td>
</tr>
<tr>
<td>TcpKaInt</td>
<td>38</td>
<td>TCP Keepalive Interval, NUMBER.</td>
</tr>
<tr>
<td>TcpKaGbF</td>
<td>39</td>
<td>TCP Keepalive Garbage, NUMBER.</td>
</tr>
<tr>
<td>NISdmain</td>
<td>40</td>
<td>NIS Domain name, ASCII.</td>
</tr>
<tr>
<td>NISservs</td>
<td>41</td>
<td>List of NIS servers, IP.</td>
</tr>
<tr>
<td>NTPservs</td>
<td>42</td>
<td>List of NTP servers, IP.</td>
</tr>
<tr>
<td>NetBNms</td>
<td>44</td>
<td>List of NetBIOS Name servers, IP.</td>
</tr>
<tr>
<td>NetBDsts</td>
<td>45</td>
<td>List of NetBIOS Distribution servers, IP.</td>
</tr>
<tr>
<td>NetBNdT</td>
<td>46</td>
<td>NetBIOS Node type (1=B-node, 2=P, 4=M, 8=H)</td>
</tr>
<tr>
<td>NetBScop</td>
<td>47</td>
<td>NetBIOS scope, ASCII.</td>
</tr>
</tbody>
</table>
## dhcp_inittab(4)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XFontSrv</td>
<td>48</td>
<td>List of X Window Font servers, IP.</td>
</tr>
<tr>
<td>XDispMgr</td>
<td>49</td>
<td>List of X Window Display managers, IP.</td>
</tr>
<tr>
<td>LeaseTim</td>
<td>51</td>
<td>Lease Time Policy, (-1 = PERM), NUMBER.</td>
</tr>
<tr>
<td>Message</td>
<td>56</td>
<td>Message to be displayed on client, ASCII.</td>
</tr>
<tr>
<td>T1Time</td>
<td>58</td>
<td>Renewal (T1) time, NUMBER.</td>
</tr>
<tr>
<td>T2Time</td>
<td>59</td>
<td>Rebinding (T2) time, NUMBER.</td>
</tr>
<tr>
<td>NW_dmain</td>
<td>62</td>
<td>NetWare/IP Domain Name, ASCII.</td>
</tr>
<tr>
<td>NWIPOpts</td>
<td>63</td>
<td>NetWare/IP Options, OCTET (unknown type).</td>
</tr>
<tr>
<td>NIS+dom</td>
<td>64</td>
<td>NIS+ Domain name, ASCII.</td>
</tr>
<tr>
<td>NIS+serv</td>
<td>65</td>
<td>NIS+ servers, IP.</td>
</tr>
<tr>
<td>TFTPsrvN</td>
<td>66</td>
<td>TFTP server hostname, ASCII.</td>
</tr>
<tr>
<td>OptBootF</td>
<td>67</td>
<td>Optional Bootfile path, ASCII.</td>
</tr>
<tr>
<td>MblIPAgt</td>
<td>68</td>
<td>Mobile IP Home Agent, IP.</td>
</tr>
<tr>
<td>SMTPserv</td>
<td>69</td>
<td>Simple Mail Transport Protocol Server, IP.</td>
</tr>
<tr>
<td>POP3serv</td>
<td>70</td>
<td>Post Office Protocol (POP3) Server, IP.</td>
</tr>
<tr>
<td>NNTPserv</td>
<td>71</td>
<td>Network News Transport Proto. (NNTP) Server, IP.</td>
</tr>
<tr>
<td>WWWservs</td>
<td>72</td>
<td>Default WorldWideWeb Server, IP.</td>
</tr>
<tr>
<td>Fingersv</td>
<td>73</td>
<td>Default Finger Server, IP.</td>
</tr>
<tr>
<td>IRCservs</td>
<td>74</td>
<td>Internet Relay Chat Server, IP.</td>
</tr>
<tr>
<td>STservs</td>
<td>75</td>
<td>StreetTalk Server, IP.</td>
</tr>
<tr>
<td>STDAServs</td>
<td>76</td>
<td>StreetTalk Directory Assist. Server, IP.</td>
</tr>
<tr>
<td>UserClas</td>
<td>77</td>
<td>User class information, ASCII.</td>
</tr>
<tr>
<td>SLP_DA</td>
<td>78</td>
<td>Directory agent, OCTET.</td>
</tr>
<tr>
<td>SLP_SS</td>
<td>79</td>
<td>Service scope, OCTET.</td>
</tr>
<tr>
<td>AgentOpt</td>
<td>82</td>
<td>Agent circuit ID, OCTET.</td>
</tr>
<tr>
<td>FQDN</td>
<td>89</td>
<td>Fully Qualified Domain Name, OCTET.</td>
</tr>
<tr>
<td>PXEarch</td>
<td>93</td>
<td>Client system architecture, NUMBER.</td>
</tr>
<tr>
<td>PXEnii</td>
<td>94</td>
<td>Client Network Device Interface, OCTET.</td>
</tr>
</tbody>
</table>
EXAMPLES

EXAMPLE 1 Altering the DHCP inittab File

In general, the DHCP inittab file should only be altered to add SITE options. If other options are added, they will not be automatically carried forward when the system is upgraded. For instance:

```
ipPairs    SITE, 132, IP, 2, 0, sdmi
```

describes an option named ipPairs, that is in the SITE category. That is, it is defined by each individual site, and is option code 132, which is of type IP Address, consisting of a potentially infinite number of pairs of IP addresses.

FILES

/etc/dhcp/inittab

ATTRIBUTES

See attributes(5) for 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

dhcpinfo(1), dhcpagent(1M), isspace(3C), dhctab(4), attributes(5), dhcp(5), dhcp_modules(5)

System Administration Guide, Volume 3


The Dynamic Host Configuration Protocol (DHCP) network tables are used to map the client identifiers of DHCP clients to IP addresses and the associated configuration parameters of that address. One DHCP network table exists for each network served by the DHCP server, and each table is named using the network’s IP address. There is no table or file with the name dhcp_network.

The DHCP network tables can exist as ASCII text files, binary text files, or NIS+ tables, depending on the data store used. Since the format of the file could change, the preferred method of managing the DHCP network tables is through the use of dhcpmgr(1M) or the pntadm(1M) command.

The format of the records in a DHCP network table depends on the data store used to maintain the table. However, an entry in a DHCP network table must contain the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client_ID</td>
<td>The client identifier field, Client_ID, is an ASCII hexadecimal representation of the unique octet string value of the DHCP Client Identifier Option (code 61) which identifies a DHCP client. In the absence of the DHCP Client Identifier Option, the DHCP client is identified using the form given below for BOOTP clients. The number of characters in this field must be an even number, with a maximum length of 64 characters. Valid characters are 0 - 9 and A-F. Entries with values of 00 are freely available for dynamic allocation to requesting clients. BOOTP clients are identified by the concatenation of the network’s hardware type (as defined by RFC 1340, titled &quot;Assigned Numbers&quot;) and the client’s hardware address. For example, the following BOOTP client has a hardware type of ’01’ (10mb ethernet) and a hardware address of 8:0:20:11:12:b7, so its client identifier would be: 010800201112B7</td>
</tr>
<tr>
<td>Flags</td>
<td>The Flags field is a decimal value, the bit fields of which can have a combination of the following values: 1 (PERMANENT) Evaluation of the Lease field is turned off (lease is permanent). If this bit is not set, Evaluation of the Lease field is enabled and the Lease is DYNAMIC. 2 (MANUAL) This entry has a manual client ID binding (cannot be reclaimed by DHCP server). Client will not be allocated another address. 4 (UNUSABLE) When set, this value means that either through ICMP echo or client DECLINE, this address has been found to be unusable. Can also be used by the network administrator to prevent a certain client from booting, if used in conjunction with the MANUAL flag.</td>
</tr>
</tbody>
</table>
This entry is reserved for allocation to BOOTP clients only.

**Client_IP**
- The **Client_IP** field holds the IP address for this entry. This value must be unique in the database.

**Server_IP**
- This field holds the IP address of the DHCP server which *owns* this client IP address, and thus is responsible for initial allocation to a requesting client.

**Lease**
- This numeric field holds the entry’s absolute lease expiration time, and is in seconds since January 1, 1970. It can be decimal, or hexadecimal (if 0x prefixes number). The special value -1 is used to denote a permanent lease.

**Macro**
- This ASCII text field contains the **dhcptab** macro name used to look up this entry’s configuration parameters in the **dhcptab** database.

**Comment**
- This ASCII text field contains an optional comment.

---

**TREATISE ON LEASES**

This section describes how the DHCP/BOOTP server calculates a client’s configuration lease using information contained in the **dhcptab** and DHCP network tables. The server consults the **LeaseTim** and **LeaseNeg** symbols in the **dhcptab**, and the **Flags** and **Lease** fields of the chosen IP address record in the DHCP network table.

The server first examines the **Flags** field for the identified DHCP network table record. If the **PERMANENT** flag is on, then the client’s lease is considered permanent.

If the **PERMANENT** flag is not on, the server checks if the client’s lease as represented by the **Lease** field in the network table record has expired. If the lease is not expired, the server checks if the client has requested a new lease. If the **LeaseNeg** symbol has not been included in the client’s **dhcptab** parameters, then the client’s requested lease extension is ignored, and the lease is set to be the time remaining as shown by the **Lease** field. If the **LeaseNeg** symbol has been included, then the server will extend the client’s lease to the value it requested if this requested lease is less than or equal to the current time plus the value of the client’s **LeaseTim** **dhcptab** parameter.

If the client’s requested lease is greater than policy allows (value of **LeaseTim**), then the client is given a lease equal to the current time plus the value of **LeaseTim**. If **LeaseTim** is not set, then the default **LeaseTim** value is one hour.

For more information about the **dhcptab** symbols, see **dhcptab**(4).

---

**SEE ALSO**

- **dhcpconfig**(1M), **dhcpmgr**(1M), **dhtadm**(1M), **in.dhcpd**(1M), **pntadm**(1M), **dhcptab**(4), **dhcp**(5), **dhcp_modules**(5)
- **Solaris DHCP Service Developer’s Guide**
- **System Administration Guide, Volume 3**
**NAME**
dhcpsvc.conf – file containing service configuration parameters for the DHCP service

**DESCRIPTION**
The dhcpsvc.conf file resides in directory `/etc/inet` and contains parameters for specifying Dynamic Host Configuration Protocol (DHCP) service configuration settings, including the type and location of DHCP data store used.

The description of the dhcpsvc.conf file in this man page is informational only. The preferred method of setting or modifying values within the dhcpsvc.conf file is by using dhcpconfig(1M) or the dhcppmgr(1M) utility. Do not edit the dhcpsvc.conf file.

The dhcpsvc.conf file format is ASCII; comment lines begin with the crosshatch (#) character. Parameters consist of a keyword followed by an equals (=) sign followed by the parameter value, of the form:

Keyword=Value

The following `Keyword` and `Value` parameters are supported:

**BOOTP_COMPAT**
String. automatic or manual. Enables support of BOOTP clients. Default is no BOOTP. Value selects BOOTP address allocation method. automatic to support all BOOTP clients, manual to support only registered BOOTP clients. server mode only parameter.

**CACHE_TIMEOUT**
Integer. Number of seconds the server will cache data from data store. Used to improve performance. Default is 10 seconds. server mode only parameter.

**CONVER**
Integer. Container version. Used by DHCP administrative tools to identify which version of the public module is being used to administer the data store. CONVER should not be changed manually.

**DAEMON_ENABLED**
`TRUE/FALSE`. If TRUE, the DHCP daemon can be run. If FALSE, DHCP daemon process will exit immediately if the daemon is started. Default is `TRUE`. Generic parameter.

**HOSTS_DOMAIN**
String. Defines name service domain that DHCP administration tools use when managing the hosts table. Valid only when HOSTS_RESOURCE is set to nisplus or dns.

**HOSTS_RESOURCE**
String. Defines what name service resource should be used by the DHCP administration tools when managing the hosts table. Current valid values are files, nisplus, and dns.

**ICMP_VERIFY**
`TRUE/FALSE`. Toggles ICMP echo verification of IP addresses. Default is `TRUE`. server mode only parameter.
INTERFACES
   String. Comma-separated list of interface names to listen to. Generic parameter.

LOGGING_FACILITY
   Integer. Local facility number (0–7 inclusive) to log DHCP events to. Default is not to log transactions. Generic parameter.

OFFER_CACHE_TIMEOUT
   Integer. Number of seconds before OFFER cache timeouts occur. Default is 10 seconds. server mode only parameter.

PATH
   Path to DHCP data tables within the data store specified by the RESOURCE parameter. The value of the PATH keyword is specific to the RESOURCE.

RELAY_DESTINATIONS
   String. Comma-separated list of host names and/or IP addresses of relay destinations. relay mode only parameter.

RELAY_HOPS
   Integer. Max number of BOOTP relay hops before packet is dropped. Default is 4. Generic parameter.

RESCAN_INTERVAL
   Integer. Number of minutes between automatic dhcptab rescans. Default is not to do rescans. server mode only parameter.

RESOURCE
   Data store resource used. Use this parameter to name the public module. See the PATH keyword in dhcp_modules(5).

RESOURCE_CONFIG
   String. This might be used for a database account name or other authentication or authorization parameters required by a particular data store. dhcp_modules(5).

Providers can use the RESOURCE_CONFIG known as configure by specifying an optional service provider layer API function:

```c
int configure(const char *configp);
```

If this function is defined by the public module provider, it is called during module load time by the private layer, with the contents of the RESOURCE_CONFIG string acquired by the administrative interface (in the case of the dhcpmgr, through the use of a public module-specific java bean extending the dhcpmgr to provide a configuration dialog for this information.

RUN_MODE
   server or relay. Selects daemon run mode. Default is server.

SECONDARY_SERVER_TIMEOUT
   Integer. The number of seconds a secondary server will wait for a primary server to respond before responding itself. Default is 20 seconds. This is a server mode only parameter.
UPDATE_TIMEOUT
Integer. Number of minutes to wait for a response from the DNS server before timing out. If this parameter is present, the DHCP daemon will update DNS on behalf of DHCP clients, and will wait the number of seconds specified for a response before timing out. You can use UPDATE_TIMEOUT without specifying a number to enable DNS updates with the default timeout of 15 minutes. If this parameter is not present, the DHCP daemon will not update DNS for DHCP clients.

VERBOSE
TRUE/FALSE. Toggles verbose mode, determining amount of status and error messages reported by the daemon. Default is FALSE. Set to TRUE only for debugging. Generic parameter.

SEE ALSO
dhcpmgr(1M), in.dhcpd(1M), dhcp(5), dhcp_modules(5)

System Administration Guide, Volume 3
NAME | dhcptab – DHCP configuration parameter table

DESCRIPTION
The dhcptab configuration table allows network administrators to organize groups of configuration parameters as macro definitions, which can then be referenced in the definition of other useful macros. These macros are then used by the DHCP server to return their values to DHCP and BOOTP clients.

The preferred method of managing the dhcptab is through the use of the dhcpmgr(1M) or dhtadm(1M) utility. The description of dhcptab entries included in this manual page is intended for informational purposes only, and should not be used to manually edit entries.

You can view the contents of the dhcptab using the DHCP manager’s tabs for Macros and Options, or using the dhtadm -P command.

Syntax of dhcptab Entries
The format of a dhcptab table depends on the data store used to maintain it. However, any dhcptab must contain the following fields in each record:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>This field identifies the macro or symbol record and is used as a search key into the dhcptab table. The name of a macro or symbol must consist of ASCII characters, with the length limited to 128 characters. Names can include spaces, except at the end of the name. The name is not case-sensitive.</td>
</tr>
<tr>
<td>Type</td>
<td>This field specifies the type of record and is used as a search key into the dhcptab. Currently, there are only two legal values for Type:</td>
</tr>
<tr>
<td></td>
<td>m – This record is a DHCP macro definition.</td>
</tr>
<tr>
<td></td>
<td>s – This record is a DHCP symbol definition. It is used to define vendor and site-specific options.</td>
</tr>
<tr>
<td>Value</td>
<td>This field contains the value for the specified type of record. For the m type, the value will consist of a series of symbol=value pairs, separated by the colon (:) character. For the s type, the value will consist of a series of fields, separated by a comma (,), which define a symbol’s characteristics. Once defined, a symbol can be used in macro definitions.</td>
</tr>
</tbody>
</table>

Symbol Characteristics
The Value field of a symbols definition contain the following fields describing the characteristics of a symbol:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context</td>
<td>This field defines the context in which the symbol definition is to be used. It can have one of the following values:</td>
</tr>
<tr>
<td></td>
<td>Site – This symbol defines a site-specific option, codes 128-254.</td>
</tr>
</tbody>
</table>
|        | Vendor=Client Class... – This symbol defines a vendor-specific option, codes 1-254. The Vendor context takes ASCII string arguments which identify the
client class that this vendor option is associated with. Multiple client class names can be specified, separated by white space. Only those clients whose client class matches one of these values will see this option. For Sun machines, the Vendor client class matches the value returned by the command `uname -i` on the client, with periods replacing commas.

**Code**
This field specifies the option code number associated with this symbol. Valid values are 128-254 for site-specific options, and 1-254 for vendor-specific options.

**Type**
This field defines the type of data expected as a value for this symbol, and is not case-sensitive. Legal values are:

- **ASCII**
  NVT ASCII text. Value is enclosed in double-quotes ("). Granularity setting has no effect on symbols of this type, since ASCII strings have a natural granularity of one (1).

- **BOOLEAN**
  No value is associated with this data type. Presence of symbols of this type denote boolean TRUE, whereas absence denotes FALSE. Granularity and Maximum values have no meaning for symbols of this type.

- **IP**
  Dotted decimal form of an Internet address. Multi-IP address granularity is supported.

- **NUMBER**
  An unsigned number with a supported granularity of 1, 2, 4, and 8 octets.
  Valid NUMBER types are: UNUMBER8, SNUMBER8, UNUMBER16, SNUMBER16, UNUMBER32, SNUMBER32, UNUMBER64, and SNUMBER64. See `dhcp_inittab(4)` for details.

- **OCTET**
  Uninterpreted ASCII representation of binary data. The client identifier is one example of an OCTET string. Valid characters are 0–9, [a-f] [A-F]. One ASCII character represents one nibble (4 bits), thus two ASCII characters are needed to represent an 8 bit quantity. The granularity setting has no effect on symbols of this type, since OCTET strings have a natural granularity of one (1).

**Granularity**
This value specifies how many objects of Type define a single instance of the symbol value. For example, the static route option is defined to be a variable list of routes. Each route consists of two IP addresses, so the Type is defined to be IP, and the data’s
granularity is defined to be 2 IP addresses. The granularity field affects the IP and NUMBER data types.

**Maximum**

This value specifies the maximum items of Granularity which are permissible in a definition using this symbol. For example, there can only be one IP address specified for a subnet mask, so the Maximum number of items in this case is one (1). A Maximum value of zero (0) means that a variable number of items is permitted.

The following example defines a site-specific option (symbol) called MystatRt, of code 130, type IP, and granularity 2, and a Maximum of 0. This definition corresponds to the internal definition of the static route option (StaticRt).

```
MystatRt = Site,130,IP,2,0
```

**Macro Definitions**

The following example illustrates a macro defined using the MystatRt site option symbol just defined:

```
10netnis m :MystatRt=3.0.0.0 10.0.0.30:
```

Macros can be specified in the Macro field in DHCP network tables (see `dhcp_network(4)`), which will bind particular macro definitions to specific IP addresses.

Up to four macro definitions are consulted by the DHCP server to determine the options that are returned to the requesting client.

These macros are processed in the following order:

**Client Class**

A macro named using the ASCII representation of the client class (e.g. SUNW.Ultra-30) is searched for in the dhcptab. If found, its symbol/value pairs will be selected for delivery to the client. This mechanism permits the network administrator to select configuration parameters to be returned to all clients of the same class.

**Network**

A macro named by the dotted Internet form of the network address of the client’s network (for example, 10.0.0.0) is searched for in the dhcptab. If found, its symbol/value pairs will be combined with those of the Client Class macro. If a symbol exists in both macros, then the Network macro value overrides the value defined in the Client Class macro. This mechanism permits the network administrator to select configuration parameters to be returned to all clients on the same network.

**IP Address**

This macro may be named anything, but must be specified in the DHCP network table for the IP address
record assigned to the requesting client. If this macro is found in the dhcptab, then its symbol/value pairs will be combined with those of the Client Class macro and the Network macro. This mechanism permits the network administrator to select configuration parameters to be returned to clients using a particular IP address. It can also be used to deliver a macro defined to include “server-specific” information by including this macro definition in all DHCP network table entries owned by a specific server.

Client Identifier

A macro named by the ASCII representation of the client’s unique identifier as shown in the DHCP network table (see dhcp_network(4)). If found, its symbol/value pairs are combined to the sum of the Client Class, Network, and IP Address macros. Any symbol collisions are replaced with those specified in the client identifier macro. The client mechanism permits the network administrator to select configuration parameters to be returned to a particular client, regardless of what network that client is connected to.

Refer to System Administration Guide, Volume 3 for more information about macro processing.

Refer to the dhcp_inittab(4) man page for more information about symbols used in Solaris DHCP.

SEE ALSO

dhcptab(4), dhcpmgr(1M), dhdtadm(1M), in.dhcpd(1M), dhcp_inittab(4), dhcp_network(4), dhcp(5)

System Administration Guide, Volume 3


Droms, R., Interoperation Between DHCP and BOOTP, RFC 1534, Bucknell University, October 1993.


dialups – list of terminal devices requiring a dial-up password

SYNOPSIS

`/etc/dialups`

DESCRIPTION

dialups is an ASCII file which contains a list of terminal devices that require a dial-up password. A dial-up password is an additional password required of users who access the computer through a modem or dial-up port. The correct password must be entered before the user is granted access to the computer. The set of ports that require a dial-up password are listed in the dialups file.

Each entry in the dialups file is a single line of the form:

```
terminal-device
```

where

`terminal-device`

The full path name of the terminal device that will require a dial-up password for users accessing the computer through a modem or dial-up port.

The dialups file should be owned by the root user and the root group. The file should have read and write permissions for the owner (root) only.

EXAMPLES

EXAMPLE 1 A sample dialups file.

Here is a sample dialups file:

```
/dev/term/a
/dev/term/b
/dev/term/c
```

FILES

```
/etc/d_passwd      dial-up password file
/etc/dialups      list of dial-up ports requiring dial-up passwords
```

SEE ALSO

d_passwd(4)
A directory consists of some number of blocks of \texttt{DIRBLKSIZ} bytes, where \texttt{DIRBLKSIZ} is chosen such that it can be transferred to disk in a single atomic operation (for example, 512 bytes on most machines).

Each \texttt{DIRBLKSIZ}-byte block contains some number of directory entry structures, which are of variable length. Each directory entry has a \texttt{struct direct} at the front of it, containing its inode number, the length of the entry, and the length of the name contained in the entry. These entries are followed by the name padded to a 4 byte boundary with null bytes. All names are guaranteed null-terminated. The maximum length of a name in a directory is \texttt{MAXNAMLEN}.

```c
#define DIRBLKSIZ DEV_BSIZE
#define MAXNAMLEN 256
struct direct {
    ulong_t d_ino; /* inode number of entry */
    ushort_t d_reclen; /* length of this record */
    ushort_t d_namlen; /* length of string in d_name */
    char d_name[MAXNAMLEN + 1]; /* maximum name length */
};
```

### 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>Stability Level</td>
<td>Unstable</td>
</tr>
</tbody>
</table>

### SEE ALSO
fs_ufs(4), attributes(5)
d_passwd – dial-up password file

/etc/d_passwd

A dial-up password is an additional password required of users who access the computer through a modem or dial-up port. The correct password must be entered before the user is granted access to the computer.

`d_passwd` is an ASCII file which contains a list of executable programs (typically shells) that require a dial-up password and the associated encrypted passwords. When a user attempts to log in on any of the ports listed in the `dialups` file (see `dialups(4)`), the login program looks at the user's login entry stored in the `passwd` file (see `passwd(4)`), and compares the login shell field to the entries in `d_passwd`. These entries determine whether the user will be required to supply a dial-up password.

Each entry in `d_passwd` is a single line of the form:

```
login-shell:password:
```

where

- **login-shell**: The name of the login program that will require an additional dial-up password.
- **password**: A 13-character encrypted password. Users accessing the computer through a dial-up port or modem using `login-shell` will be required to enter this password before gaining access to the computer.

`d_passwd` should be owned by the `root` user and the `root` group. The file should have read and write permissions for the owner (`root`) only.

If the user's login program in the `passwd` file is not found in `d_passwd` or if the login shell field in `passwd` is empty, the user must supply the default password. The default password is the entry for `/usr/bin/sh`. If `d_passwd` has no entry for `/usr/bin/sh`, then those users whose login shell field in `passwd` is empty or does not match any entry in `d_passwd` will not be prompted for a dial-up password.

Dial-up logins are disabled if `d_passwd` has only the following entry:

```
/usr/bin/sh:*:
```

**EXAMPLE 1** Sample `d_passwd` file.

Here is a sample `d_passwd` file:

```
/usr/lib/uucp/uucico:q.mJzTnu8icF0:
/usr/bin/csh:6k/7KCFRPNVXg:
/usr/bin/ksh:9df/FDf.4jkRt:
/usr/bin/sh:41FuGVzGcDJlw:
```
Generating An Encrypted Password

The passwd (see passwd(1)) utility can be used to generate the encrypted password for each login program. passwd generates encrypted passwords for users and places the password in the shadow (see shadow(4)) file. Passwords for the d_passwd file will need to be generated by first adding a temporary user id using useradd (see useradd(1M)), and then using passwd(1) to generate the desired password in the shadow file. Once the encrypted version of the password has been created, it can be copied to the d_passwd file.

For example:

1. Type useradd tempuser and press Return. This creates a user named tempuser.
2. Type passwd tempuser and press Return. This creates an encrypted password for tempuser and places it in the shadow file.
3. Find the entry for tempuser in the shadow file and copy the encrypted password to the desired entry in the d_passwd file.
4. Type userdel tempuser and press Return to delete tempuser.

These steps must be executed as the root user.

FILES
/etc/d_passwd
/etc/dialups
/etc/passwd
/etc/shadow

SEE ALSO
passwd(1), useradd(1M), dialups(4), passwd(4), shadow(4)

WARNINGS

When creating a new dial-up password, be sure to remain logged in on at least one terminal while testing the new password. This ensures that there is an available terminal from which you can correct any mistakes that were made when the new password was added.
Driver configuration files pass information about device drivers and their configuration to the system. Most device drivers do not have to have configuration files. Drivers for devices that are self-identifying, such as the SBus devices on many systems, can usually obtain all the information they need from the FCode PROM on the SBus card using the DDI property interfaces. See `ddi_prop_get_int(9F)` and `ddi_prop_lookup(9F)` for details.

The system associates a driver with its configuration file by name. For example, a driver in `/usr/kernel/drv` called `wombat` has the driver configuration file `wombat.conf` associated with it. By convention, the driver configuration file lives in the same directory as the driver.

The syntax of a single entry in a driver configuration file takes one of three forms:

```
name="node name" parent="parent name"  [property-name=value ...];
```

In this form, the parent name can be either a simple nexus driver name to match all instances of that parent/node, or the parent name can be a specific full pathname, beginning with a slash (/) character, identifying a specific instance of a parent bus.

Alternatively, the parent can be specified by the type of interface it presents to its children.

```
name="node name" class="class name"  [property-name=value ...];
```

For example, the driver for the SCSI host adapter may have different names on different platforms, but the target drivers can use class `scsi` to insulate themselves from these differences.

Entries of either form above correspond to a device information (devinfo) node in the kernel device tree. Each node has a `name` which is usually the name of the driver, and a `parent` name which is the name of the parent devinfo node it will be connected to. Any number of name-value pairs may be specified to create properties on the prototype devinfo node. These properties can be retrieved using the DDI property interfaces (for example, `ddi_prop_get_int(9F)` and `ddi_prop_lookup(9F)`). The prototype devinfo node specification must be terminated with a semicolon (`;`).

The third form of an entry is simply a list of properties.

```
[property-name=value ...];
```

A property created in this way is treated as global to the driver. It can be overridden by a property with the same name on a particular devinfo node, either by creating one explicitly on the prototype node in the driver.conf file or by the driver.

Items are separated by any number of newlines, SPACE or TAB characters.
The configuration file may contain several entries to specify different device configurations and parent nodes. The system may call the driver for each possible prototype devinfo node, and it is generally the responsibility of the drivers probe(9E) routine to determine if the hardware described by the prototype devinfo node is really present.

Property names should obey the same naming convention as Open Boot PROM properties, in particular they should not contain at-sign (@), or slash (/) characters. Property values can be decimal integers or strings delimited by double quotes ("). Hexadecimal integers can be constructed by prefixing the digits with 0x.

A comma separated list of integers can be used to construct properties whose value is an integer array. The value of such properties can be retrieved inside the driver using ddi_prop_lookup_int_array(9F).

Comments are specified by placing a # character at the beginning of the comment string, the comment string extends for the rest of the line.

EXAMPLES

EXAMPLE 1 Configuration file for a PCI bus frame buffer.

The following is an example of a configuration file called ACME,simple.conf for a PCI bus frame buffer called ACME,simple.

```
# # Copyright (c) 1993, by ACME Fictitious Devices, Inc.
# #ident @(#)ACME,simple.conf 1.3  1999/09/09

name="ACME,simple" class="pci" unit-address="3,1"
  debug-mode=12;
```

This example creates a prototype devinfo node called ACME,simple under all parent nodes of class pci. It specifies a property called reg that consists of an array of three integers. The reg property is interpreted by the parent node; see pci(4) for further details.

EXAMPLE 2 Configuration file for a pseudo device driver

The following is an example of a configuration file called ACME,example.conf for a pseudo device driver called ACME,example.

```
# # Copyright (c) 1993, ACME Fictitious Devices, Inc.
# #ident @(#)ACME,example.conf 1.2  93/09/09

name="ACME,example" parent="pseudo" instance=0
driver.conf(4)
driver.conf(4)
driver.conf(4)

    debug-level=1;
```

```

    name="ACME,example" parent="pseudo" instance=1;
```

```

    whizzy-mode="on";
```

```

    debug-level=3;
```

```
EXAMPLE 2 Configuration file for a pseudo device driver  (Continued)

This creates two devinfo nodes called ACME, example which will attach below the pseudo node in the kernel device tree. The instance property is only interpreted by the pseudo node, see pseudo(4) for further details. A property called debug-level will be created on the first devinfo node which will have the value 1. The example driver will be able to fetch the value of this property using ddi_prop_get_int(9F).

Two global driver properties are created, whizzy-mode (which will have the string value "on") and debug-level (which will have the value 3). If the driver looks up the property whizzy-mode on either node, it will retrieve the value of the global whizzy-mode property ("on"). If the driver looks up the debug-level property on the first node, it will retrieve the value of the debug-level property on that node (1). Looking up the same property on the second node will retrieve the value of the global debug-level property (3).

SEE ALSO  pci(4), pseudo(4), sbus(4), scsi(4), pci(4), probe(9E), ddi_getlongprop(9F), ddi_getprop(9F), ddi_getproplen(9F), ddi_prop_op(9F)

Writing Device Drivers

WARNINGS  To avoid namespace collisions between multiple driver vendors, it is strongly recommended that the name property of the driver should begin with a vendor-unique string. A reasonably compact and unique choice is the vendor over-the-counter stock symbol.
environ(4)

NAME     environ, pref, variables – user-preference variables files for AT&T FACE

SYNOPSIS

$HOME/pref/.environ
$HOME/pref/.variables
$HOME/FILECABINET/.pref
$HOME/WASTEBASKET/.pref

DESCRIPTION

The .environ, .pref, and .variables files contain variables that indicate user preferences for a variety of operations. The .environ and .variables files are located under the user's $HOME/pref directory. The .pref files are found under $HOME/FILECABINET, $HOME/WASTEBASKET, and any directory where preferences were set via the organize command. Names and descriptions for each variable are presented below. Variables are listed one per line and are of the form `variable=value`.

### .environ Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGINWIN[1-4]</td>
<td>Windows that are opened when FACE is initialized.</td>
</tr>
<tr>
<td>SORTMODE</td>
<td>Sort mode for file folder listings. Values include the following hexadecimal digits:</td>
</tr>
<tr>
<td></td>
<td>1 Sorted alphabetically by name.</td>
</tr>
<tr>
<td></td>
<td>2 Files most recently modified first.</td>
</tr>
<tr>
<td></td>
<td>800 Sorted alphabetically by object type.</td>
</tr>
<tr>
<td></td>
<td>The values above may be listed in reverse order by ORing the following value:</td>
</tr>
<tr>
<td></td>
<td>1000 List objects in reverse order. For example, a value of 1002 will produce a folder listing with files LEAST recently modified displayed first. A value of 1001 would produce a &quot;reverse&quot; alphabetical by name listing of the folder.</td>
</tr>
<tr>
<td>DISPLAYMODE</td>
<td>Display mode for file folders. Values include the following hexadecimal digits:</td>
</tr>
<tr>
<td></td>
<td>0 File names only.</td>
</tr>
<tr>
<td></td>
<td>4 File names and brief description.</td>
</tr>
<tr>
<td></td>
<td>8 File names, description, plus additional information.</td>
</tr>
<tr>
<td>WASTEPROMPT</td>
<td>Prompt before emptying wastebasket (yes/no?).</td>
</tr>
<tr>
<td>WASTEDAYS</td>
<td>Number of days before emptying wastebasket.</td>
</tr>
<tr>
<td>PRINCMD[1-3]</td>
<td>Print command defined to print files.</td>
</tr>
<tr>
<td>UMASK</td>
<td>Holds default permissions with which files will be created.</td>
</tr>
</tbody>
</table>
**.pref Variables**  
Variables found in `.pref` are the following:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SORTMODE</td>
<td>Contains the same values as the SORTMODE variable described in <code>.environ</code> above.</td>
</tr>
<tr>
<td>DISPLAYMODE</td>
<td>Contains the same values as the DISPLAYMODE variable described in <code>.environ</code> above.</td>
</tr>
</tbody>
</table>

**.variable Variables**  
Variables found in `.variables` include:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDITOR</td>
<td>Default editor.</td>
</tr>
<tr>
<td>PS1</td>
<td>Shell prompt.</td>
</tr>
</tbody>
</table>
**ethers(4)**

<table>
<thead>
<tr>
<th>NAME</th>
<th>ethers – Ethernet address to hostname database or domain</th>
</tr>
</thead>
</table>
| DESCRIPTION | The *ethers* file is a local source of information about the (48 bit) Ethernet addresses of hosts on the Internet. The *ethers* file can be used in conjunction with or instead of other ethers sources, including the NIS maps *ethers(byname)* and *ethers(byaddr)* and the NIS+ table *ethers*. Programs use the *ethers*(3SOCKET) routines to access this information. The *ethers* file has one line for each host on an Ethernet. The line has the following format:

```
Ethernet-address official-host-name
```

Items are separated by any number of SPACE and/or TAB characters. A ‘#’ indicates the beginning of a comment extending to the end of line.

The standard form for Ethernet addresses is “x:x:x:x:x:x” where x is a hexadecimal number between 0 and ff, representing one byte. The address bytes are always in network order. Host names may contain any printable character other than SPACE, TAB, NEWLINE, or comment character. |
| FILES | /etc/ethers |
| SEE ALSO | ethers(3SOCKET), hosts(4), nsswitch.conf(4) |
### exec_attr(4)

**NAME**
exec_attr – execution profiles database

**SYNOPSIS**
/etc/security/exec_attr

**DESCRIPTION**
/etc/security/exec_attr is a local database that specifies the execution attributes associated with profiles. The exec_attr file can be used with other sources for execution profiles, including the exec_attr NIS map and NIS+ table. Programs use the getexecattr(3SECDB) routines to access this information.

The search order for multiple execution profile sources is specified in the /etc/nsswitch.conf file, as described in the nsswitch.conf(4) man page. The search order follows the entry for prof_attr(4).

A profile is a logical grouping of authorizations and commands that is interpreted by a profile shell to form a secure execution environment. The shells that interpret profiles are pfcsh, pfksh, and pfsh. See the pfsh(1) man page. Each user’s account is assigned zero or more profiles in the user_attr(4) database file.

Each entry in the exec_attr database consists of one line of text containing seven fields separated by colons (:). Line continuations using the backslash (\) character are permitted. The basic format of each entry is:

```
name:policy:type:res1:res2:id:attr
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>name</strong></td>
<td>The name of the profile. Profile names are case-sensitive.</td>
</tr>
<tr>
<td><strong>policy</strong></td>
<td>The policy that is associated with the profile entry. The only valid policy is suser.</td>
</tr>
<tr>
<td><strong>type</strong></td>
<td>The type of object defined in the profile. The only valid type is cmd.</td>
</tr>
<tr>
<td><strong>res1</strong></td>
<td>Reserved for future use.</td>
</tr>
<tr>
<td><strong>res2</strong></td>
<td>Reserved for future use.</td>
</tr>
<tr>
<td><strong>id</strong></td>
<td>A string that uniquely identifies the object described by the profile. For a profile of type cmd, the id is either the full path to the command or the asterisk (*) symbol, which is used to allow all commands. An asterisk that replaces the filename component in a pathname indicates all files in a particular directory. To specify arguments, the pathname should point to a shell script written to execute the command with the desired arguments.</td>
</tr>
<tr>
<td><strong>attr</strong></td>
<td>An optional list of semicolon-separated (;) key-value pairs that describe the security attributes to apply to the object upon execution. Zero or more keys may be specified. The list of valid key words depends on the policy enforced. The following key words are valid: euid, uid, egid, and gid.</td>
</tr>
</tbody>
</table>
euid and uid contain a single user name or a numeric user ID. Commands designated with euid run with the effective UID indicated, which is similar to setting the setuid bit on an executable file. Commands designated with uid run with both the real and effective UIDs. Setting uid may be more appropriate than setting the euid on privileged shell scripts.

egid and gid contain a single group name or a numeric group ID. Commands designated with egid run with the effective GID indicated, which is similar to setting the setgid bit on a file. Commands designated with gid run with both the real and effective GIDs. Setting gid may be more appropriate than setting egid on privileged shell scripts.

EXAMPLE 1 Using effective user and group IDs

The following example shows the audit command specified in the Audit Control profile to execute with an effective user ID of root (0) and effective group ID of bin (3):

Audit Control:suser:cmd:::/etc/init.d/audit:euid=0;egid=3

FILES
/etc/nsswitch.conf
/etc/user_attr
/etc/security/exec_attr

CAVEATS
When deciding which authorization source to use (see DESCRIPTION), keep in mind that NIS+ provides stronger authentication than NIS.

Because the list of legal keys is likely to expand, any code that parses this database must be written to ignore unknown key-value pairs without error. When any new keywords are created, the names should be prefixed with a unique string, such as the company’s stock symbol, to avoid potential naming conflicts.

The following characters are used in describing the database format and must be escaped with a backslash if used as data: colon (:), semicolon (;), equals (=), and backslash (\).

SEE ALSO
auths(1), profiles(1), roles(1), makedbm(1M), getauthattr(3SECDB), getauusernam(3BSM), getexecattr(3SECDB), getprofattr(3SECDB), getuserattr(3SECDB), kva_match(3SECDB), auth_attr(4), prof_attr(4), user_attr(4)
These files, conventionally called /dev/fd/0, /dev/fd/1, /dev/fd/2, and so on, refer to files accessible through file descriptors. If file descriptor n is open, these two system calls have the same effect:

```c
fd = open("/dev/fd/n",mode);
fd = dup(n);
```

On these files creat(2) is equivalent to open, and mode is ignored. As with dup, subsequent reads or writes on fd fail unless the original file descriptor allows the operations.

For convenience in referring to standard input, standard output, and standard error, an additional set of names is provided: /dev/stdin is a synonym for /dev/fd/0, /dev/stdout for /dev/fd/1, and /dev/stderr for /dev/fd/2.

SEE ALSO: creat(2), dup(2), open(2)

DIAGNOSTICS: open(2) returns -1 and EBADF if the associated file descriptor is not open.
A flash archive is an easily transportable version of a reference configuration of the Solaris operating environment, plus optional other software. Such an archive is used for the rapid installation of Solaris on large numbers of machines. The machine that contains a flash archive is referred to as a master system. A machine that receives a copy of a flash archive is called a clone system.

Flash archives are monolithic files containing both archive identification information and the actual files that have been copied from a master system and that will be extracted onto a clone system.

The flash archive is laid out in the following sections:

- archive cookie
- archive identification
- user-defined (optional)
- archive files

The only assumptions regarding section number and placement that an application processing the archive can make is that there is an identification section located immediately after the archive cookie and that the last section in the archive is an archive files section.

These sections are described in the following subsections.

**Archive Cookie**

The very beginning of the archive contains a cookie, which serves to identify the file as a flash archive. It is also used by the deployment code for identification and validation purposes.

The case-sensitive, newline-terminated cookie that identifies version 1.0 flash archives, is FlAsH-arChIve-1.0.

The archive version is designed to allow for the future evolution of the flash archive specification while allowing applications that process flash archives to determine whether specific archives are of a format that can be handled correctly. The archive version is a number of the form x.y, where x is the major version number, and y is the minor version number.

The major and/or minor version numbers must be incremented when changes are made to the archive specification. The type and impact of the changes being made determine which of the version numbers must be incremented. A minor change, indicated by an incrementing of the minor version number, is one that will not negatively impact an application’s ability to extract the archive. A major change, indicated by an incrementing of the major version number, is the converse. When an application encounters a flash archive with an unknown major version number, it should issue an error message and exit.
The archive identification section is plain text, delimited with newline characters. It is composed of a series of keyword/value pairs, with one pair allowed per line. Keywords and values are separated by a single equal sign. There are no limits to the length of individual lines. Binary data to be included as the value to a keyword is base64 encoded. The keywords themselves are case-insensitive. The case-sensitivity of the values is determined by the definition of the keyword, though most are case-insensitive.

The global order of the keywords within the identification section is undefined, save for the section boundary keywords. The identification section must begin with section_begin=ident and must end with section_end=ident.

In addition to the keywords defined for the flash archive and enumerated below, users can define their own. These user-defined keywords are ignored by the flash mechanisms, but can be used by user-provided scripts or programs that process the identification section. User-defined keywords must begin with X, and contain characters other than linefeeds, equal signs, and null characters. For example, X-department is a valid user-defined keyword. department, which lacks the X-prefix, is not. Suggested naming conventions for user-defined keyword include the underscore-delimited descriptive method used for the pre-defined keywords, or a federated convention similar to that used to name Java packages.

Applications that process the identification section will process unrecognized non-user-defined keyword differently, depending on whether the archive version is known. If the application recognizes the archive specification version, it will reject any unrecognized non-user-defined keyword. If the application does not recognize the specification version, that is, if the minor version number is higher than the highest minor version it knows how to process, unrecognized non-user-defined keywords will be ignored. These ignored keyword are reported to the user by means of a non-fatal warning message.

The keywords defined for this version of the Flash archive specification are listed below.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Value</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>section_begin</td>
<td>text</td>
<td>yes</td>
</tr>
<tr>
<td>section_end</td>
<td>text</td>
<td>yes</td>
</tr>
<tr>
<td>archive_id</td>
<td>text</td>
<td>no</td>
</tr>
<tr>
<td>files_archived_method</td>
<td>text</td>
<td>no</td>
</tr>
<tr>
<td>files_compressed_method</td>
<td>text</td>
<td>no</td>
</tr>
<tr>
<td>files_archived_size</td>
<td>numeric</td>
<td>no</td>
</tr>
<tr>
<td>files_unarchived_size</td>
<td>numeric</td>
<td>no</td>
</tr>
</tbody>
</table>
Note that future versions of the identification section might define additional keywords. The only guarantee regarding the new keywords is that they will not intrude upon the user-defined keyword namespace as given above.

The following is an example identification section:

```
section_begin=identification
files_archived_method=cpio
files_compressed_method=compress
files_archived_size=259323342
files_unarchived_size=591238111
creation_date=20000131221409
creation_master=pumbaa
content_name=Finance Print Server
content_type=server
content_description=Solaris 8 Print Server
content_author=Mighty Matt
content_architectures=sun4u,sun4m
x-department=Internal Finance
```

The following are descriptions of the identification section keywords:

```plaintext
section_begin
section_end
```

These keywords are used to delimit sections in the archive and are not limited exclusively to the identification section. For example, the archive files section includes a section_begin keyword, though with a different value. User-defined archive sections will be delimited by section_begin and section_end keywords, with values appropriate to each section. The currently defined section names are given in the table below. User-defined names should follow the same convention as user-defined identification sections, with the additional restriction that they not contain forward slashes (/).
Note that while the archive cookie does not use section boundaries, and thus has no need for a section name within the archive itself, the `flar(1M)` command uses section names when splitting the archive, and thus requires a section name for the archive cookie. The name `cookie` is reserved for that purpose.

The following four keywords, used in the archive identification section, describe the contents of the archive files section.

archive_id

This optional keyword `uniquely` describes the contents of the archive. It is computed as a unique hash value of the bytes representing the archive. Currently this value is represented as an ASCII hexadecimal 128-bit MD5 hash of the archive contents. This value is used by the installation software only to validate the contents of the archive during archive installation.

If the keyword is present, the hash value is recomputed during extraction based on the contents of the archive being extracted. If the recomputed value does not match the stored value in the identification section, then the archive is deemed corrupt, and appropriate actions can be taken by the application.

If the keyword is not present, then no integrity check is performed.

files_archived_method

This keyword describes the archive method used in the files section. If this keyword is not present, the files section is assumed to be in CPIO format with ASCII headers (the `-c` option to `cpio`). If the keyword is present, it can have the following value:

`cpio` The archive format in the files section is CPIO with ASCII headers.

The compression method indicated by the `files_compressed_method` keyword (if present) is applied to the archive file created by the archive method.

The introduction of additional archive methods will require a change in the major archive specification version number, as applications aware only of `cpio` will be unable to extract archives that use other archive methods.

`files_compressed_method`
This keyword describes the compression algorithm (if any) used on the files section. If this keyword is not present, the files section is assumed to be uncompressed. If the keyword is present, it can have one of the following values:

- **none**  The files section is not compressed.
- **compress**  The files section is compressed using `compress(1)`.

The compression method indicated by this keyword is applied to the archive file created by the archive method indicated by the value of the `files_archived_method` keyword (if any). **gzip** compression of the flash archive is not currently supported, as the `gzip` decompression program is not included in the standard miniroot.

Introduction of an additional compression algorithm would require a change in the major archive specification version number, as applications aware only of the above methods will be unable to extract archives that use other compression algorithms.

### `files_archived_size`

The value associated with this keyword is the size of the archived files section, in bytes. This value is used by the deployment software only to give extraction progress information to the user. While the deployment software can easily determine the size of the archived files section prior to extraction, it cannot do so in the case of archive retrieval via a stream. To determine the compressed size when extracting from a stream, the extraction software would have to read the stream twice. This double read would result in an unacceptable performance penalty compared to the value of the information gathered.

If the keyword is present, the value is used only for the provision of status information. Because this keyword is only advisory, deployment software must be able to handle extraction of archives for which the actual file section size does not match the size given in `files_archive_size`.

If `files_archive_size` is not present and the archive is being read from a stream device that does not allow the prior determination of size information, such as a tape drive, completion status information will not be generated. If the keyword is not present and the archive is being read from a random-access device such as a mounted filesystem, or from a stream that provides size information, the compressed size will be generated dynamically and used for the provision of status information.

### `files_unarchived_size`

This keyword defines the cumulative size in bytes of the extracted archive. The value is used for filesystem size verification. The following verification methods are possible using this approach:

- **No checking**  If the `files_unarchived_size` keyword is absent, no spacechecking will be performed.
Aggregate checking

If the `files_unarchived_size` keyword is present and the associated value is an integer, the integer will be compared against the aggregate free space created by the requested filesystem configuration.

The following keywords provide descriptive information about the archive as a whole. They are generally used to assist the user in archive selection and to aid in archive management. These keywords are all optional and are used by the deployment programs only to assist the user in distinguishing between individual archives.

**creation_date**

The value of the `creation_date` keyword is a textual timestamp representing the time of creation for the archive. The value of this keyword can be overridden at archive creation time through the `flarcreate(1M)`. The timestamp must be in ISO-8601 complete basic calendar format without the time designator (ISO-8601, §5.4.1(a)) as follows:

```
CCYYMMDDhhmmss
```

For example:

```
20000131221409
(January 31st, 2000 10:14:09pm)
```

The date and time included in the value should be in GMT.

**creation_master**

The value of the `creation_master` keyword is the name of the master machine used to create the archive. The value can be overridden at archive creation time.

**content_name**

The value of the `content_name` keyword should describe the archive’s function and purpose. It is similar to the `NAME` parameter found in Solaris packages.

The value of the `content_name` keyword is used by the deployment utilities to identify the archive and can be presented to the user during the archive selection process and/or the extraction process. The value must be no longer than 256 characters.

**content_type**

The value of this keyword specifies a category for the archive. This category is defined by the user and is used by deployment software for display purposes. This keyword is the flash analog of the Solaris packaging `CATEGORY` keyword.

**content_description**
The value of this keyword is used to provide the user with a description of what the archive contains and should build on the description provided in content_name. In this respect, content_description is analogous to the DESC keyword used in Solaris packages.

There is no length limit to the value of content_description. To facilitate display, the value can contain escaped newline characters. As in C, the escaped newline takes the form of \n. Due to the escaped newline, backslashes must be included as \\.

The description is displayed in a non-proportional font, with at least 40 characters available per line. Lines too long for display are wrapped.

content_author

The value of this keyword is a user-defined string identifying the creator of the archive. Suggested values include the full name of the creator, the creator’s email address, or both.

content_architectures

The value of this keyword is a comma-delimited list of the kernel architectures supported by the given archive. The value of this keyword is generated at archive creation time, and can be overridden by the user at that time. If this keyword is present in the archive, the extraction mechanism validates the kernel architecture of the clone system with the list of architectures supported by the archive. The extraction fails if the kernel architecture of the clone is not supported by the archive. If the keyword is not present, no architecture validation is performed.

User-Defined Sections

Following the identification section may be zero or more user-defined sections. These sections are not processed by the archive extraction code and can be used for any purpose.

User-defined sections must be line-oriented, terminated with newline (ASCII 0x0a) characters. There is no limit on the length of individual lines. If binary data is to be included in a user-defined section, it should be encoded using base64 or a similar algorithm.

Archive Files Section

The archive files section contains the files gathered from the master system. While the length of this section should be the same as the value of the files_archived_size keyword in the identification section, you should not assume that these two values are equal. This section begins with section_begin=archive, but it does not have an ending section boundary.

ATTRIBUTES

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWinst</td>
</tr>
</tbody>
</table>
SEE ALSO  compress(1), cpio(1), flar(1M), flarcreate(1M), md5(3EXT), attributes(5)
format.dat – disk drive configuration for the format command

format.dat enables you to use your specific disk drives with format(1M). On Solaris 2.3 and compatible systems, format will automatically configure and label SCSI drives, so that they need not be defined in format.dat. Three things can be defined in the data file:

- search paths
- disk types
- partition tables.

Syntax

The following syntax rules apply to the data file:

- The pound # sign is the comment character. Any text on a line after a pound sign is not interpreted by format.
- Each definition in the format.dat file appears on a single logical line. If the definition is more than one line long, all but the last line of the definition must end with a backslash (\).
- A definition consists of a series of assignments that have an identifier on the left side and one or more values on the right side. The assignment operator is the equal sign (=). Assignments within a definition must be separated by a colon (:).
- White space is ignored by format(1M). If you want an assigned value to contain white space, enclose the entire value in double quotes ("'). This will cause the white space within quotes to be preserved as part of the assignment value.
- Some assignments can have multiple values on the right hand side. Separate values by a comma (,).

Keywords

The data file contains disk definitions that are read in by format(1M) when it starts up. Each definition starts with one of the following keywords: search_path, disk_type, and partition.

search_path

4.x: Tells format which disks it should search for when it starts up. The list in the default data file contains all the disks in the GENERIC configuration file. If your system has disks that are not in the GENERIC configuration file, add them to the search_path definition in your data file. The data file can contain only one search_path definition. However, this single definition lets you specify all the disks you have in your system.

5.x: By default, format(1M) understands all the logical devices that are of the form /dev/rdsk/cntndnsn; hence search_path is not normally defined on a 5.x system.

disk_type

Defines the controller and disk model. Each disk_type definition contains information concerning the physical geometry of the disk. The default data file contains definitions for the controllers and disks that the Solaris operating environment supports. You need to
add a new disk_type only if you have an unsupported disk. You can add as many disk_type definitions to the data file as you want.

The following controller types are supported by format(1M):

<table>
<thead>
<tr>
<th>Controller Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XY450</td>
<td>Xylogics 450 controller (SMD)</td>
</tr>
<tr>
<td>XD7053</td>
<td>Xylogics 7053 controller (SMD)</td>
</tr>
<tr>
<td>MD21</td>
<td>SCSI, but using ESDI devices (also known as shoebox)</td>
</tr>
<tr>
<td>SCSI</td>
<td>True SCSI (CCS or SCSI-2)</td>
</tr>
<tr>
<td>ISP-80</td>
<td>IPI panther controller</td>
</tr>
</tbody>
</table>

Note: The disk_type and partition definition entries must have "ctlr = MD21" for scsi disk devices for 4.1.1 release. But for 4.1.2, 4.1.3 and 5.x releases, the entries should say "ctlr = SCSI."

The keyword itself is assigned the name of the disk type. This name appears in the disk's label and is used to identify the disk type whenever format(1M) is run. Enclose the name in double quotes to preserve any white space in the name.

Below are lists of identifiers for supported controllers. Note that an asterisk (*) indicates the identifier is mandatory for that controller -- it is not part of the keyword name.

The following identifiers are assigned values in all disk_type definitions:

- acyl* alternate cylinders
- asect alternate sectors per track
- atrks alternate tracks
- fmt_time formatting time per cylinder
- ncl* number of logical cylinders
- nhead* number of logical heads
- nsect* number of logical sectors per track
- pcyl* number of physical cylinders
- phead number of physical heads
- psect number of physical sectors per track
- rpm* drive RPM
These identifiers are for SCSI and MD-21 Controllers

- `read_retries` page 1 byte 3 (read retries)
- `write_retries` page 1 byte 8 (write retries)
- `cyl_skew` page 3 bytes 18-19 (cylinder skew)
- `trk_skew` page 3 bytes 16-17 (track skew)
- `trks_zone` page 3 bytes 2-3 (tracks per zone)
- `cache` page 38 byte 2 (cache parameter)
- `prefetch` page 38 byte 3 (prefetch parameter)
- `max_prefetch` page 38 byte 4 (minimum prefetch)
- `min_prefetch` page 38 byte 6 (maximum prefetch)

Note: The Page 38 values are device-specific. Refer the user to the particular disk’s manual for these values.

For SCSI disks, the following geometry specifiers may cause a mode select on the byte(s) indicated:

- `asect` page 3 bytes 4-5 (alternate sectors per zone)
- `atrks` page 3 bytes 8-9 (alt. tracks per logical unit)
- `phead` page 4 byte 5 (number of heads)
- `psect` page 3 bytes 10-11 (sectors per track)

And these identifiers are for SMD Controllers Only

- `bps*` bytes per sector (SMD)
- `bpt*` bytes per track (SMD)

Note: under SunOS 5.x, bpt is only required for SMD disks. Under SunOS 4.x, bpt was required for all disk types, even though it was only used for SMD disks.

And this identifier is for XY450 SMD Controllers Only

- `drive_type*` drive type (SMD) (just call this "xy450 drive type")

**partition** Defines a partition table for a specific disk type. The partition table contains the partitioning information, plus a name that lets you refer to it in format(1M). The default data file contains default partition definitions for several kinds of disk drives. Add a partition definition if you repartitioned any of the disks on your system. Add as many partition definitions to the data file as you need.
Partition naming conventions differ in SunOS 4.x and in SunOS 5.x.

4.x: the partitions are named as a, b, c, d, e, f, g, h.

5.x: the partitions are referred to by numbers 0, 1, 2, 3, 4, 5, 6, 7.

**EXAMPLE 1**

A sample disk_type and partition.

Following is a sample disk_type and partition definition in format.dat file for SUN0535 disk device.

```plaintext
disk_type = "SUN0535" \\
    : ctlr = SCSI : fmt_time = 4 \\
    : ncyl = 1866 : acyl = 2 : pcyl = 2500 : nhead = 7 : nsect = 80 \\
    : rpm = 5400 
partition = "SUN0535" \\
    : disk = "SUN0535" : ctlr = SCSI \\
    : 0 = 0, 64400 : 1 = 115, 103600 : 2 = 0, 1044960 : 6 = 300, 876960 
```

**FILES**

/etc/format.dat default data file if format -x is not specified, nor is there a format.dat file in the current directory.

**SEE ALSO**

format(1M) *System Administration Guide, Volume 1*
NAME
fspec – format specification in text files

DESCRIPTION
It is sometimes convenient to maintain text files on the system with non-standard tabs, (tabs that are not set at every eighth column). Such files must generally be converted to a standard format, frequently by replacing all tabs with the appropriate number of spaces, before they can be processed by system commands. A format specification occurring in the first line of a text file specifies how tabs are to be expanded in the remainder of the file.

A format specification consists of a sequence of parameters separated by blanks and surrounded by the brackets < : and : >. Each parameter consists of a keyletter, possibly followed immediately by a value. The following parameters are recognized:

- **tabs**
  - The **tabs** parameter specifies the tab settings for the file. The value of **tabs** must be one of the following:
    - A list of column numbers separated by commas, indicating tabs set at the specified columns.
    - A ‘−’ followed immediately by an integer **n**, indicating tabs at intervals of **n** columns.
    - A ‘−’ followed by the name of a “canned” tab specification.

  Standard tabs are specified by **t−8**, or equivalently, **t1,9,17,25**, etc. The canned tabs that are recognized are defined by the **tabs(1)** command.

- **size**
  - The **size** parameter specifies a maximum line size. The value of **size** must be an integer. Size checking is performed after tabs have been expanded, but before the margin is prepended.

- **margin**
  - The **margin** parameter specifies a number of spaces to be prepended to each line. The value of **margin** must be an integer.

- **d**
  - The **d** parameter takes no value. Its presence indicates that the line containing the format specification is to be deleted from the converted file.

- **e**
  - The **e** parameter takes no value. Its presence indicates that the current format is to prevail only until another format specification is encountered in the file.

Default values, which are assumed for parameters not supplied, are **t−8** and **m0**. If the **s** parameter is not specified, no size checking is performed. If the first line of a file does not contain a format specification, the above defaults are assumed for the entire file. The following is an example of a line containing a format specification:

```
* <:t5,10,15 s72:> *
```

If a format specification can be disguised as a comment, it is not necessary to code the **d** parameter.
SEE ALSO  ed(1), newform(1), tabs(1)
NAME
fstypes – file that registers distributed file system packages

DESCRIPTION
fstypes resides in directory /etc/dfs and lists distributed file system utilities packages installed on the system. For each installed distributed file system type, there is a line that begins with the file system type name (for example, “nfs”), followed by white space and descriptive text.

The file system indicated in the first line of the file is the default file system; when Distributed File System (DFS) Administration commands are entered without the option −F fstypes, the system takes the file system type from the first line of the fstypes file.

The default file system can be changed by editing the fstypes file with any supported text editor.

SEE ALSO
dfmounts(1M), dfshares(1M), share(1M), shareall(1M), unshare(1M)
**NAME**

fs_ufs, inode_ufs, inode – format of a ufs file system volume

**SYNOPSIS**

```c
#include <sys/param.h>
#include <sys/types.h>
#include <sys/fs/ufs_fs.h>
#include <sys/fs/ufs_inode.h>
```

**DESCRIPTION**

Standard UFS file system storage volumes have a common format for certain vital information. Every volume is divided into a certain number of blocks. The block size is a parameter of the file system. Sectors 0 to 15 contain primary and secondary bootstrapping programs.

The actual file system begins at sector 16 with the super-block. The layout of the super-block is defined by the header `<sys/fs/ufs_fs.h>`.

Each disk drive contains some number of file systems. A file system consists of a number of cylinder groups. Each cylinder group has inodes and data.

A file system is described by its super-block, and by the information in the cylinder group blocks. The super-block is critical data and is replicated before each cylinder group block to protect against catastrophic loss. This is done at file system creation time and the critical super-block data does not change, so the copies need not be referenced.

**fs_clean**

`fs_clean` indicates the state of the file system. The `FSCLEAN` state indicates an undamaged, cleanly unmounted file system. The `FSACTIVE` state indicates a mounted file system that has been updated. The `FSSTABLE` state indicates an idle mounted file system. The `FSFIX` state indicates that this fs is mounted, contains inconsistent file system data and is being repaired by `fsck`. The `FSBAD` state indicates that this file system contains inconsistent file system data. It is not necessary to run `fsck` on any unmounted file systems with a state of `FSCLEAN` or `FSSTABLE`. `mount(2)` will return `ENOSPC` if a UFS file system with a state of `FSACTIVE` is being mounted for read-write.

To provide additional safeguard, `fs_clean` could be trusted only if `fs_state` contains a value equal to `FSOKAY - fs_time`, where `FSOKAY` is a constant integer. Otherwise, `fs_clean` is treated as though it contains the value of `FSACTIVE`.

Addresses stored in inodes are capable of addressing fragments of “blocks.” File system blocks of at most, size `MAXBSIZE` can be optionally broken into 2, 4, or 8 pieces, each of which is addressable; these pieces may be `DEV_BSIZE` or some multiple of a `DEV_BSIZE` unit.
Large files consist exclusively of large data blocks. To avoid undue wasted disk space, the last data block of a small file is allocated only as many fragments of a large block as are necessary. The file system format retains only a single pointer to such a fragment, which is a piece of a single large block that has been divided. The size of such a fragment is determinable from information in the inode, using the `blksize(fs, ip, lbn)` macro.

The file system records space availability at the fragment level; aligned fragments are examined to determine block availability.

The root inode is the root of the file system. Inode 0 cannot be used for normal purposes and historically, bad blocks were linked to inode 1. Thus the root inode is 2 (inode 1 is no longer used for this purpose; however numerous dump tapes make this assumption, so we are stuck with it). The `lost+found` directory is given the next available inode when it is initially created by `mkfs(1M).

`fs_minfree` gives the minimum acceptable percentage of file system blocks which may be free. If the freelist drops below this level only the super-user may continue to allocate blocks. `fs_minfree` may be set to 0 if no reserve of free blocks is deemed necessary, however severe performance degradations will be observed if the file system is run at greater than 90% full; thus the default value of `fs_minfree` is 10%.

Empirically the best trade-off between block fragmentation and overall disk utilization at a loading of 90% comes with a fragmentation of 8; thus the default fragment size is an eighth of the block size.

`fs_optim` specifies whether the file system should try to minimize the time spent allocating blocks, or if it should attempt to minimize the space fragmentation on the disk. If the value of `fs_minfree` is less than 10%, then the file system defaults to optimizing for space to avoid running out of full sized blocks. If the value of `fs_minfree` is greater than or equal to 10%, fragmentation is unlikely to be problematical, and the file system defaults to optimizing for time.

Cylinder group related limits: Each cylinder keeps track of the availability of blocks at different rotational positions, so that sequential blocks can be laid out with minimum rotational latency. `fs_nrpos` is the number of rotational positions which are distinguished. With the default `fs_nrpos` of 8, the resolution of the summary information is 2ms for a typical 3600 rpm drive.

`fs_rotdelay` gives the minimum number of milliseconds to initiate another disk transfer on the same cylinder. It is used in determining the rotationally optimal layout for disk blocks within a file; the default value for `fs_rotdelay` varies from drive to drive (see `tunefs(1M)`).

`fs_maxcontig` gives the maximum number of blocks, belonging to one file, that will be allocated contiguously before inserting a rotational delay.

Each file system has a statically allocated number of inodes. An inode is allocated for each `NBPI` bytes of disk space. The inode allocation strategy is extremely conservative.
MINBSIZE is the smallest allowable block size. With a MINBSIZE of 4096 it is possible to create files of size $2^{32}$ with only two levels of indirection. MINBSIZE must be large enough to hold a cylinder group block, thus changes to (struct cg) must keep its size within MINBSIZE. Note: super-blocks are never more than size SBSIZE.

The path name on which the file system is mounted is maintained in fs_fsmnt. MAXMNTLEN defines the amount of space allocated in the super-block for this name.

The limit on the amount of summary information per file system is defined by MAXCSBUFS. It is currently parameterized for a maximum of two million cylinders.

Per cylinder group information is summarized in blocks allocated from the first cylinder group’s data blocks. These blocks are read in from _fs_csaddr_ (size _fs_cssize_) in addition to the super-block.

Note: sizeof (struct csum) must be a power of two in order for the _fs_cs_ macro to work.

The inode is the focus of all file activity in the file system. There is a unique inode allocated for each active file, each current directory, each mounted-on file, text file, and the root. An inode is “named” by its device/i-number pair. For further information, see the header <sys/fs/ufs_inode.h>.

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>Stability Level</td>
<td>Unstable</td>
</tr>
</tbody>
</table>

SEE ALSO fsck_ufs(1M), mkfs_ufs(1M), tunefs(1M), mount(2), attributes(5)
ftpusers(4)

NAME     ftpusers – file listing users to be disallowed ftp login privileges
SYNOPSIS /etc/ftpusers

DESCRIPTION The /etc/ftpusers is an ASCII file that lists users for whom ftp login privileges are disallowed. Each ftpuser entry is a single line of the form:

name

where name is the user’s login name.

The ftp server, in.ftpd(1M), reads the ftpusers file. If the login name of the user matches one of the entries listed, it rejects the login session and sends the Login incorrect and Login failed error messages.

The ftpusers file has the following default configuration entries:

- root
- daemon
- bin
- sys
- adm
- lp
- uucp
- nuucp
- listen
- nobody
- noaccess
- nobody4

These entries match the default instantiated entries from passwd(4). The list of default entries typically contains the superuser root and other administrative and system application identities.

The root entry is included in /etc/ftpusers as a security measure since the default policy is to disallow remote logins for this identity. This policy is also set in the the default value of the CONSOLE entry in the /etc/default/login file. See login(1).

If you allow root login privileges by deleting the root entry in /etc/ftpusers, you should also should modify the security policy in /etc/default/login to reflect the site security policy for remote login access by root.

Other default entries are administrative identities that are typically assumed by system applications but never used for local or remote login, for example sys and nobody. Since these entries do not have a valid password field instantiated in shadow(4), no login can be performed.

If a site adds similar administrative or system application identities in passwd(4) and shadow(4), for example, majoromo, the site should consider including them in /etc/ftpusers for a consistent security policy.

FILES /etc/ftpusers
       /etc/default/login
       /etc/passwd
       /etc/shadow
See attributes(5) for descriptions of the following attributes:

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

See also login(1), in.ftpd(1M), passwd(4), shadow(4), attributes(5), environ(5)
An input file to `geniconvtbl` is an ASCII text file that contains an iconv code conversion definition from one codeset to another codeset.

The `geniconvtbl` utility accepts the code conversion definition file(s) and writes code conversion binary table file(s) that can be used in `iconv(1)` and `iconv(3C)` to support user-defined code conversions. See `iconv(1)` and `iconv(3C)` for more detail on the iconv code conversion and `geniconvtbl(1)` for more detail on the utility.

The following lexical conventions are used in the iconv code conversion definition:

- **CONVERSION_NAME**: A string of characters representing the name of the iconv code conversion. The iconv code conversion name should start with one or more printable ASCII characters followed by a percentage character '%', followed by another one or more of printable ASCII characters. Examples: `ISO8859-1%ASCII`, `646%eucJP`, `CP_939%ASCII`.

- **NAME**: A string of characters starts with any one of the ASCII alphabet characters or the underscore character, '_' followed by one or more ASCII alphanumeric characters and underscore character, '_'. Examples: `_a1, ABC_codeset, K1`.

- **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, 0X1A, 0x1B3`.

- **DECIMAL**: A decimal number, represented by one or more decimal digits. Examples: `0, 123, 2165`.

Each comment starts with `//` and ends at the end of the line.

The following keywords are reserved:

- automatic
- between
- binary
- break
- condition
- default
- dense
- direction
- discard
- else
- error
- escapeseq
- false
- if
- index
- init
- input
- inputsize
Additionally, the following symbols are also reserved as tokens:

```plaintext
[];....
```

The following table shows the precedence and associativity of the operators from lower precedence at the top to higher precedence at the bottom of the table allowed in the iconv code conversion definition:

<table>
<thead>
<tr>
<th>Operator (Symbol)</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignment (=)</td>
<td>Right</td>
</tr>
<tr>
<td>Logical OR (</td>
<td></td>
</tr>
<tr>
<td>Logical AND (&amp;&amp;)</td>
<td>Left</td>
</tr>
<tr>
<td>Bitwise OR (</td>
<td>)</td>
</tr>
<tr>
<td>Exclusive OR (^)</td>
<td>Left</td>
</tr>
<tr>
<td>Bitwise AND (&amp;)</td>
<td>Left</td>
</tr>
<tr>
<td>Equal-to (= =),</td>
<td>Left</td>
</tr>
<tr>
<td></td>
<td>Inequality (!=)</td>
</tr>
<tr>
<td>Less-than (&lt;),</td>
<td>Left</td>
</tr>
<tr>
<td></td>
<td>Less-than-or-equal-to (&lt;=),</td>
</tr>
<tr>
<td></td>
<td>Greater-than (&gt;),</td>
</tr>
<tr>
<td></td>
<td>Greater-than-or-equal-to (&gt;=)</td>
</tr>
<tr>
<td>Left-shift (&lt;&lt;),</td>
<td>Left</td>
</tr>
<tr>
<td></td>
<td>Right-shift (&gt;&gt;)</td>
</tr>
<tr>
<td>Addition (+),</td>
<td>Left</td>
</tr>
<tr>
<td></td>
<td>Subtraction (-)</td>
</tr>
<tr>
<td>Multiplication (*),</td>
<td>Left</td>
</tr>
<tr>
<td></td>
<td>Division (/),</td>
</tr>
<tr>
<td></td>
<td>Remainder (%)</td>
</tr>
</tbody>
</table>

File Formats 135
<table>
<thead>
<tr>
<th>Logical negation (!), Right</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitwise complement (~),</td>
</tr>
<tr>
<td>Unary minus (-)</td>
</tr>
</tbody>
</table>

### The Syntax

Each iconv code conversion definition starts with `CONVERSION_NAME` followed by one or more semi-colon separated code conversion definition elements:

```c
// a US-ASCII to ISO8859-1 iconv code conversion example:
US-ASCII%ISO8859-1 {
    // one or more code conversion definition elements here.
    :
    :
}
```

Each code conversion definition element can be any one of the following elements:

- **direction**
- **condition**
- **operation**
- **map**

To have a meaningful code conversion, there should be at least one direction, operation, or map element in the iconv code conversion definition.

The direction element contains one or more semi-colon separated condition-action pairs that direct the code conversion:

```c
direction For_US-ASCII_2_ISO8859-1 {
    // one or more condition-action pairs here.
    :
    :
}
```

Each condition-action pair contains a conditional code conversion that consists of a condition element and an action element.

```c
condition action
```

If the pre-defined condition is met, the corresponding action is executed. If there is no pre-defined condition met, iconv(3C) will return -1 with errno set to EILSEQ. The condition can be a condition element, a name to a pre-defined condition element, or a
condition literal value, true. The 'true' condition literal value always yields success and thus the corresponding action is always executed. The action also can be an action element or a name to a pre-defined action element.

The condition element specifies one or more condition expression elements. Since each condition element can have a name and also can exist stand-alone, a pre-defined condition element can be referenced by the name at any action pairs later. To be used in that way, the corresponding condition element should be defined beforehand:

```
condition For_US-ASCII_2_ISO8859-1 {
   // one or more condition expression elements here.
}
```

The name of the condition element in the above example is `For_US-ASCII_2_ISO8859-1`. Each condition element can have one or more condition expression elements. If there are more than one condition expression elements, the condition expression elements are checked from top to bottom to see if any one of the condition expression elements will yield a true. Any one of the following can be a condition expression element:

- `between`
- `escapeseq`
- `expression`

The `between` condition expression element defines one or more comma-separated ranges:

```
between 0x0...0x1f, 0x7f...0x9f ;
between 0xa1a1...0xfe ;
```

In the first expression in the example above, the covered ranges are `0x0` to `0x1f` and `0x7f` to `0x9f` inclusively. In the second expression, the covered range is the range whose first byte is `0xa1` to `0xfe` and whose second byte is between `0xa1` to `0xfe`. This means that the range is defined by each byte. In this case, the sequence `0xa280` does not meet the range.

The `escapeseq` condition expression element defines an equal-to condition for one or more comma-separated escape sequence designators:

```
// ESC $ ) C sequence:
escapeseq 0x1b242943;
// ESC $ ) C sequence or ShiftOut (SO) control character code, 0x0e:
escapeseq 0x1b242943, 0x0e;
```

The expression can be any one of the following and can be surrounded by a pair of parentheses, '(' and ')':

geniconvtbl(4)
// HEXADECIMAL: 
0xa1a1

// DECIMAL
12

// A boolean value, true:
true

// A boolean value, false:
false

// Addition expression:
1 + 2

// Subtraction expression:
10 - 3

// Multiplication expression:
0x20 * 10

// Division expression:
20 / 10

// Remainder expression:
17 % 3

// Left-shift expression:
1 << 4

// Right-shift expression:
0xa1 >> 2

// Bitwise OR expression:
0x2121 | 0x8080

// Exclusive OR expression:
0xa1a1 ^ 0x8080

// Bitwise AND expression:
0xa1 & 0x80

// Equal-to expression:
0x10 == 16

// Inequality expression:
0x10 != 10

// Less-than expression:
0x20 < 25

// Less-than-or-equal-to expression:
10 <= 0x10

// Bigger-than expression:
0x10 > 12

// Bigger-than-or-equal-to expression:
There is a single type available in this expression: integer. The boolean values are two special cases of integer values. The ‘true’ boolean value’s integer value is 1 and the ‘false’ boolean value’s integer value is 0. Also, any integer value other than 0 is a true boolean value. Consequently, the integer value 0 is the false boolean value. Any boolean expression yields integer value 1 for true and integer value 0 for false as the result.

Any literal value shown at the above expression examples as operands, that is, DECIMAL, HEXADECIMAL, and boolean values, can be replaced with another expression. There are a few other special operands that you can use as well in the expressions: ‘input’, ‘inputsize’, ‘outputsize’, and variables. input is a keyword pointing to the current input buffer. inputsize is a keyword pointing to the current input buffer size in bytes. outputsize is a keyword pointing to the current output buffer size in bytes. The NAME lexical convention is used to name a variable. The initial value of a variable is 0. The following expressions are allowed with the special operands:

// Pointer to the third byte value of the current input buffer:
input[2]

// Equal-to expression with the ‘input’:
input == 0x8020

// Alternative way to write the above expression:
0x8020 == input

// The size of the current input buffer size:
inputsize

// The size of the current output buffer size:
outputsize

// A variable:
saved_second_byte

// Assignment expression with the variable:
saved_second_byte = input[1]
The input keyword without index value can be used only with the equal-to operator, `==`. When used in that way, the current input buffer is consecutively compared with another operand byte by byte. An expression can be another operand. If the input keyword is used with an index value \( n \), it is a pointer to the \((n+1)\)th byte from the beginning of the current input buffer. An expression can be the index. Only a variable can be placed on the left hand side of an assignment expression.

The action element specifies an action for a condition and can be any one of the following elements:

- direction
- operation
- map

The operation element specifies one or more operation expression elements:

```plaintext
operation For_US-ASCII_2_ISO8859-1 {
    // one or more operation expression element definitions here.
    ;
    ;
}
```

If the name of the operation element, in the case of the above example, `For_US-ASCII_2_ISO8859-1`, is either `init` or `reset`, it defines the initial operation and the reset operation of the `iconv` code conversion:

```plaintext
// The initial operation element:
operation init {
    // one or more operation expression element definitions here.
    ;
    ;
}

// The reset operation element:
operation reset {
    // one or more operation expression element definitions here.
    ;
    ;
}
```

The initial operation element defines the operations that need to be performed in the beginning of the `iconv` code conversion. The reset operation element defines the operations that need to be performed when a user of the `iconv(3)` function requests a state reset of the `iconv` code conversion. For more detail on the state reset, refer to `iconv(3C)`.
The operation expression can be any one of the following three different expressions and each operation expression should be separated by an ending semicolon:

if-else operation expression
output operation expression
control operation expression

The if-else operation expression makes a selection depend on the boolean expression result. If the boolean expression result is true, the true task that follows the ‘if’ is executed. If the boolean expression yields false and if a false task is supplied, the false task that follows the ‘else’ is executed. There are three different kinds of if-else operation expressions:

// The if-else operation expression with only true task:
if (expression) {
    // one or more operation expression element definitions here.
    ;
    ;
}

// The if-else operation expression with both true and false tasks:
if (expression) {
    // one or more operation expression element definitions here.
    ;
    ;
} else {
    // one or more operation expression element definitions here.
    ;
    ;
}

// The if-else operation expression with true task and another if-else operation expression as the false task:
if (expression) {
    // one or more operation expression element definitions here.
    ;
    ;
} else if (expression) {
    // one or more operation expression element definitions here.
    ;
    ;
} else {
    // one or more operation expression element definitions here.
The last if-else operation expression can have another if-else operation expression as the false task. The other if-else operation expression can be any one of above three if-else operation expressions.

The output operation expression saves the right hand side expression result to the output buffer:

```c
// Save 0x8080 at the output buffer:
output = 0x8080;
```

If the size of the output buffer left is smaller than the necessary output buffer size resulting from the right hand side expression, the iconv code conversion will stop with E2BIG errno and `(size_t)-1` return value to indicate that the code conversion needs more output buffer to complete. Any expression can be used for the right hand side expression. The output buffer pointer will automatically move forward appropriately once the operation is executed.

The control operation expression can be any one of the following expressions:

```c
// Return (size_t)-1 as the return value with an EINVAL errno:
error;

// Return (size_t)-1 as the return value with an EBADF errno:
error 9;

// Discard input buffer byte operation. This discards a byte from
// the current input buffer and move the input buffer pointer to
// the 2'nd byte of the input buffer:
discard;

// Discard input buffer byte operation. This discards
// 10 bytes from the current input buffer and move the input
// buffer pointer to the 11'th byte of the input buffer:
discard 10;

// Return operation. This stops the execution of the current
// operation:
return;

// Operation execution operation. This executes the init
// operation defined and sets all variables to zero:
operation init;

// Operation execution operation. This executes the reset
// operation defined and sets all variables to zero:
operation reset;

// Operation execution operation. This executes an operation
// defined and named 'ISO8859_1_to_ISO8859_2':
```
operation ISO8859_1_to_ISO8859_2;

// Direction operation. This executes a direction defined and
// named 'ISO8859_1_to_KOI8_R';
direction ISO8859_1_to_KOI8_R;

// Map execution operation. This executes a mapping defined
// and named 'Map_ISO8859_1_to_US_ASCII';
map Map_ISO8859_1_to_US_ASCII;

// Map execution operation. This executes a mapping defined
// and named 'Map_ISO8859_1_to_US_ASCII' after discarding
// 10 input buffer bytes;
map Map_ISO8859_1_to_US_ASCII 10;

In case of init and reset operations, if there is no pre-defined init and/or reset
operations in the iconv code conversions, only system-defined internal init and reset
operations will be executed. The execution of the system-defined internal init and reset
operations will clear the system-maintained internal state.

There are three special operators that can be used in the operation:

printchr expression;
printhd expression;
printint expression;

The above three operators will print out the given expression as a character, a
hexadecimal number, and a decimal number, respectively, at the standard error
stream. These three operators are for debugging purposes only and should be
removed from the final version of the iconv code conversion definition file.

In addition to the above operations, any valid expression separated by a semi-colon
can be an operation, including an empty operation, denoted by a semi-colon alone as
an operation.

The map element specifies a direct code conversion mapping by using one or more
map pairs. When used, usually many map pairs are used to represent an iconv code
conversion definition:

map For_US-ASCII_2_ISO8859-1 {

    // one or more map pairs here
    ;
}

Each map element also can have one or two comma-separated map attribute elements
like the following examples:

// Map with densely encoded mapping table map type:
map maptype = dense {

// Map with hash mapping table map type with hash factor 10. // Only hash mapping table map type can have hash factor. If // the hash factor is specified with other map types, it will be // ignored.
map maptype = hash : 10 {
    // one or more map pairs here.
    :
    :
}

// Map with binary search tree based mapping table map type:
map maptype = binary {
    // one or more map pairs here.
    :
    :
}

// Map with index table based mapping table map type:
map maptype = index {
    // one or more map pairs here.
    :
    :
}

// Map with automatic mapping table map type. If defined, // system will assign the best possible map type.
map maptype = automatic {
    // one or more map pairs here.
    :
    :
}

// Map with output_byte_length limit set to 2.
map output_byte_length = 2 {
    // one or more map pairs here.
    :
    :
}

// Map with densely encoded mapping table map type and // output_bute_length limit set to 2:
map maptype = dense, output_byte_length = 2 {
If no maptype is defined, automatic is assumed. If no output_byte_length is defined, the system figures out the maximum possible output byte length for the mapping by scanning all the possible output values in the mappings. If the actual output byte length scanned is bigger than the defined output_byte_length, the geniconvtbl utility issues an error and stops generating the code conversion binary table(s).

The following are allowed map pairs:

// Single mapping. This maps an input character denoted by
// the code value 0x20 to an output character value 0x21:
0x20 0x21

// Multiple mapping. This maps 128 input characters to 128
// output characters. In this mapping, 0x0 maps to 0x10, 0x1 maps
// to 0x11, 0x2 maps to 0x12, ..., and, 0x7f maps to 0x8f:
0x0...0x7f 0x10...0x8f

// Default mapping. If specified, every undefined input character
// in this mapping will be converted to a specified character
// (in the following case, a character with code value of 0x3f):
default 0x3f;

// Default mapping. If specified, every undefined input character
// in this mapping will not be converted but directly copied to
// the output buffer:
default no_change_copy;

// Error mapping. If specified, during the code conversion,
// if input buffer contains the byte value, in this case, 0x80,
// the iconv(3) will stop and return (size_t)-1 as the return
// value with EILSEQ set to the errno:
0x80 error;

If no default mapping is specified, every undefined input character in the mapping will be treated as an error mapping, and thus the iconv(3) will stop the code conversion and return (size_t)-1 as the return value with EILSEQ set to the errno.

The syntax of the iconv code conversion definition in extended BNF is illustrated below:

```
iconv_conversion_definition
   : CONVERSION_NAME '{' definition_element_list '}'
 |
definition_element_list
   : definition_element ';
   | definition_element_list definition_element ';
```

File Formats 145
geniconvtbl(4)

definition_element
  : direction
  | condition
  | operation
  | map
  ;

direction
  : 'direction' NAME '{' direction_unit_list '}
  | 'direction' '{' direction_unit_list '}
  ;
direction_unit_list
  : direction_unit
  | direction_unit_list direction_unit
  ;
direction_unit
  : condition action ';
  | condition NAME ';
  | NAME action ';
  | NAME NAME action ';
  | 'true' action ';
  | 'true' NAME ';
  ;
action
  : direction
  | map
  | operation
  ;
condition
  : 'condition' NAME '{' condition_list '}
  | 'condition' '{' condition_list '}
  ;
condition_list
  : condition_expr ';
  | condition_list condition_expr ';
  ;
condition_expr
  : 'between' range_list
  | expr
  | 'esccaseq' escseq_list ';
  ;
range_list
  : range_pair
  | range_list ',' range_pair
  ;
range_pair
  : HEXADECIMAL '..' HEXADECIMAL
  ;
escseq_list
: escseq
  | escseq_list ',' escseq

escseq : HEXADECIMAL

map : 'map' NAME '{' map_list '}'
  | 'map' '{' map_list '}'
  | 'map' NAME map_attribute '{' map_list '}'
  | 'map' map_attribute '{' map_list '}'

map_attribute : map_type ',', 'output_byte_length' '=' DECIMAL
  | map_type
  | 'output_byte_length' '=' DECIMAL ',' map_type
  | 'output_byte_length' '=' DECIMAL

map_type : 'maptype' '=' map_type_name : DECIMAL
  | 'maptype' '=' map_type_name

map_type_name : 'automatic'
  | 'index'
  | 'hash'
  | 'binary'
  | 'dense'

map_list : map_pair
  | map_list map_pair

map_pair : HEXADECIMAL HEXADECIMAL
  | HEXADECIMAL '...' HEXADECIMAL HEXADECIMAL
  | 'default' HEXADECIMAL
  | 'default' 'no_change_copy'
  | HEXADECIMAL 'error'

operation : 'operation' NAME '{' op_list '}'
  | 'operation' '{' op_list '}'
  | 'operation' 'init' '{' op_list '}'
  | 'operation' 'reset' '{' op_list '}'

op_list : op_unit
  | op_list op_unit

op_unit : ','
  | expr ','
  | 'error' ','
geniconvtbl(4)

'error' expr ';
'discard' ';
'discard' expr ';
'output' '=' expr ';
'direction' NAME ';
'operation' NAME ';
'operation' 'init' ';
'operation' 'reset' ';
'map' NAME ';
'map' NAME expr ';

op_if_else
'return' ';
'printchr' expr ';
'printhd' expr ';
'printint' expr ';

op_if_else
:'if' '({' expr '}') '({' op_list '}')
| 'if' '({' expr '}') '({' op_list '}') 'else' op_if_else
| 'if' '({' expr '}') '({' op_list '}') 'else' '{' op_list '}'

expr : '(' expr ')
  NAME
  HEXADECIMAL
  DECIMAL
  'input' '[' expr ']
  'outputsize'
  'inputsize'
  'true'
  'false'
  'input' '==' expr
  expr '==' 'input'
  '|' expr
  '|' expr
  '-' expr
  expr '+' expr
  expr '-' expr
  expr '*' expr
  expr '/' expr
  expr '%' expr
  expr '<=' expr
  expr '>' expr
  expr '|' expr
  expr '|=' expr
  expr '&' expr
  expr '==' expr
  expr '!=' expr
  expr '>' expr
  expr '<=' expr
  expr '>' expr
  expr '<' expr
  expr '<=' expr
  NAME '=' expr
  expr '||' expr
  expr '&&' expr

148 man pages section 4: File Formats • Last Revised 29 Oct 1999
EXAMPLE 1 Code conversion from ISO8859-1 to ISO646

ISO8859-1|ISO646 {
  // Use dense-encoded internal data structure.
  map maptype = dense {
    default 0x3f
    0x0...0x7f 0x0
  };
}

EXAMPLE 2 Code conversion from eucJP to ISO-2022-JP

// Iconv code conversion from eucJP to ISO-2022-JP

#include <sys/errno.h>

  operation init {
    codesetnum = 0;
  },

  operation reset {
    if (codesetnum != 0) {
      // Emit state reset sequence, ESC ( J, for
      // ISO-2022-JP.
      output = 0x1b284a;
    }
    operation init;
  },

  direction {
    condition { // JIS X 0201 Latin (ASCII)
      between 0x00...0x7f;
    } operation {
      if (codesetnum != 0) {
        // We will emit four bytes.
        if (outputsize <= 3) {
          error E2BIG;
        }
        // Emit state reset sequence, ESC ( J.
        output = 0x1b284a;
        codesetnum = 0;
      } else {
        if (outputsize <= 0) {
          error E2BIG;
        }
      }
      output = input[0];
      // Move input buffer pointer one byte.
      discard;
    }

    condition { // JIS X 0208
      between 0xa1a1...0xfefe;
    } operation {
      if (codesetnum != 1) {
        if (outputsize <= 4) {
error E2BIG;
}
// Emit JIS X 0208 sequence, ESC $ B.
output = 0x1b2442;
codesetnum = 1;
} else {
   if (outputsize <= 1) {
      error E2BIG;
   }
}
output = (input[0] & 0x7f);
output = (input[1] & 0x7f);

// Move input buffer pointer two bytes.
discard 2;
};
condition {
   // JIS X 0201 Kana
   between 0x8ea1...0x8edf;
} operation {
   if (codesetnum != 2) {
      if (outputsize <= 3) {
         error E2BIG;
      }
      // Emit JIS X 0201 Kana sequence,
      // ESC ( I.
      output = 0x1b2849;
codesetnum = 2;
   } else {
      if (outputsize <= 0) {
         error E2BIG;
      }
   }
   output = (input[1] & 127);
   // Move input buffer pointer two bytes.
discard 2;
};
condition {
   // JIS X 0212
   between 0x8fa1a1...0x8ffefe;
} operation {
   if (codesetnum != 3) {
      if (outputsize <= 5) {
         error E2BIG;
      }
   }
   // Emit JIS X 0212 sequence, ESC $( D.
   output = 0x1b242844;
codesetnum = 3;
} else {
   if (outputsize <= 1) {
      error E2BIG;
   }
   output = (input[1] & 127);
EXAMPLE 2 Code conversion from eucJP to ISO-2022-JP  (Continued)

```c
    output = (input[2] & 127);  
    discard 3;  
};

true operation {    // error
    error EILSEQ;
};

};
```

FILES
/usr/bin/geniconvtbl
the utility geniconvtbl
/usr/lib/iconv/geniconvtbl/binarytables/*.bt
conversion binary tables
/usr/lib/iconv/geniconvtbl/srcs/*
conversion source files for user reference

SEE ALSO
cpp(1), geniconvtbl(1), iconv(1), iconv(3C), iconv-close(3C),
icov-open(3C), attributes(5), environ(5)

International Language Environments Guide

NOTES
The maximum length of HEXADECIMAL and DECIMAL digit length is 128. The maximum length of a variable is 255. The maximum nest level is 16.
The group file is a local source of group information. The group file can be used in conjunction with other group sources, including the NIS maps group.byname and group.bygid and the NIS+ table group. Programs use the getgrnam(3C) routines to access this information.

The group file contains a one-line entry for each group recognized by the system, of the form:

```
grouplname:password: gid:user-list
```

where

- **grouplname**: The name of the group.
- **gid**: The group’s unique numerical ID (GID) within the system.
- **user-list**: A comma-separated list of users allowed in the group.

The maximum value of the gid field is 2137483647. To maximize interoperability and compatibility, administrators are recommended to assign groups using the range of GIDs below 60000 where possible.

If the password field is empty, no password is demanded. During user identification and authentication, the supplementary group access list is initialized sequentially from information in this file. If a user is in more groups than the system is configured for, [NGROUPS_MAX], a warning will be given and subsequent group specifications will be ignored.

Malformed entries cause routines that read this file to halt, in which case group assignments specified further along are never made. To prevent this from happening, use grpck(1B) to check the /etc/group database from time to time.

Previous releases used a group entry beginning with a ‘+’ (plus sign) or ‘−’ (minus sign) to selectively incorporate entries from NIS maps for group. If still required, this is supported by specifying `group:compat` in nsswitch.conf. The “compat” source may not be supported in future releases. The preferred sources are, “files” followed by “nisplus”. This has the effect of incorporating the entire contents of the NIS+ group table after the group file.

**EXAMPLE 1 Sample of a group File**

Here is a sample group file:

```
root::0:root
stooges:q.mJwTnu8icF.:10:larry,moe,curly
```

and the sample group entry from nsswitch.conf:

```
group: files nisplus
```
EXAMPLE 1 Sample of a group File (Continued)

With these entries, the group stooges will have members larry, moe, and curly, and all groups listed in the NIS+ group table are effectively incorporated after the entry for stooges.

If the group file was:

root::0::root
stooges::q.mJzTnu8icF::10::larry,moe,curly
*: 

and the group entry from nsswitch.conf:

group: compat

all the groups listed in the NIS group.bygid and group.byname maps would be effectively incorporated after the entry for stooges.

SEE ALSO

groups(1), grpck(1B), newgrp(1), getgrnam(3C), initgroups(3C),
nsswitch.conf(4), unistd(3HEAD)

System Administration Guide, Volume 1
The \texttt{/etc/acct/holidays} file describes which hours are considered prime time and which days are holidays. Holidays and weekends are considered non-prime time hours. \texttt{/etc/acct/holidays} is used by the accounting system.

All lines beginning with an "*" are comments.

The \texttt{/etc/acct/holidays} file consists of two sections. The first non-comment line defines the current year and the start time of prime and non-prime time hours, in the form:

\begin{verbatim}
 current_year prime_start non_prime_start
\end{verbatim}

The remaining non-comment lines define the holidays in the form:

\begin{verbatim}
 month/day company_holiday
\end{verbatim}

Of these two fields, only the \texttt{month/day} is actually used by the accounting system programs.

The \texttt{/etc/acct/holidays} file must be updated each year.

\textbf{EXAMPLE 1} Example of the \texttt{/etc/acct/holidays} file.

The following is an example of the \texttt{/etc/acct/holidays} file:

\begin{verbatim}
* Prime/Nonprime Table for the accounting system
* * Curr  Prime  Non-Prime
*   Year   Start  Start
  *  1991  0830    1800
* * only the first column (month/day) is significant.
* *
* month/day Company Holiday
* 1/1    New Years Day
  5/30   Memorial Day
  7/4    Independ. Day
  9/5    Labor Day
 11/24   Thanksgiving Day
 11/25   day after Thanksgiving
 12/25   Christmas
 12/26   day after Christmas
\end{verbatim}
The **hosts** file is a local database that associates the names of hosts with their Internet Protocol (IP) addresses. The **hosts** file can be used in conjunction with, or instead of, other hosts databases, including the Domain Name System (DNS), the NIS **hosts** map and the NIS+ **hosts** table. Programs use library interfaces to access information in the **hosts** file.

The **hosts** file has one entry for each IP address of each host. If a host has more than one IP address, it will have one entry for each, on consecutive lines. The format of each line is:

```
IP-address official-host-name nicknames . .
```

Items are separated by any number of SPACE and/or TAB characters. The first item on a line is the host’s IP address. The second entry is the host’s official name. Subsequent entries on the same line are alternative names for the same machine, or “nicknames.” Nicknames are optional.

For a host with more than one IP address, consecutive entries for these addresses may contain the same or differing nicknames. Different nicknames are useful for assigning distinct names to different addresses.

A call to `gethostbyname(3NSL)` returns a `hostent` structure containing the union of all addresses and nicknames from each line containing a matching official name or nickname.

A ‘#’ indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines that search the file.

Network addresses are written in the conventional “decimal dot” notation and interpreted using the `inet_addr` routine from the Internet address manipulation library, `inet(3SOCKET)`. This interface supports host names as defined in Internet RFC 952 which states:

A “name” (Net, Host, Gateway, or Domain name) is a text string up to 24 characters drawn from the alphabet (A-Z), digits (0-9), minus sign (−), and period (.). Note that periods are only allowed when they serve to delimit components of “domain style names”. (See RFC 921, Domain Name System Implementation Schedule, for background). No blank or space characters are permitted as part of a name. No distinction is made between upper and lower case. The first character must be an alpha character. The last character must not be a minus sign or period.

Although the interface accepts host names longer than 24 characters for the host portion (exclusive of the domain component), choosing names for hosts that adhere to the 24 character restriction will insure maximum interoperability on the Internet.
A host which serves as a GATEWAY should have “−GATEWAY” or “−GW” as part of its name. Hosts which do not serve as Internet gateways should not use “−GATEWAY” and “−GW” as part of their names. A host which is a TAC should have “−TAC” as the last part of its host name, if it is a DoD host. Single character names or nicknames are not allowed.

RFC 952 has been modified by RFC 1123 to relax the restriction on the first character being a digit.

EXAMPLES

EXAMPLE 1 Example of a Typical Line From the hosts File

Here is a typical line from the hosts file:

192.9.1.20 gaia # John Smith

SEE ALSO

in.named(1M), gethostbyname(3NSL), inet(3SOCKET), nsswitch.conf(4), resolv.conf(4)

NOTES

/etc/inet/hosts is the official SVR4 name of the hosts file. The symbolic link /etc/hosts exists for BSD compatibility.
hosts.equiv(4)

NAME
hosts.equiv, rhosts – trusted remote hosts and users

DESCRIPTION
The /etc/hosts.equiv and .rhosts files provide the “remote authentication”
database for rlogin(1), rsh(1), rcp(1), and rcmd(3SOCKET). The files specify remote
hosts and users that are considered “trusted”. Trusted users are allowed to access the
local system without supplying a password. The library routine ruserok() (see
rcmd(3SOCKET)) performs the authentication procedure for programs by using the
/etc/hosts.equiv and .rhosts files. The /etc/hosts.equiv file applies to the
entire system, while individual users can maintain their own .rhosts files in their
home directories.

These files bypass the standard password-based user authentication mechanism. To
maintain system security, care must be taken in creating and maintaining these files.

The remote authentication procedure determines whether a user from a remote host
should be allowed to access the local system with the identity of a local user. This
procedure first checks the /etc/hosts.equiv file and then checks the .rhosts file
in the home directory of the local user who is requesting access. Entries in these files
can be of two forms. Positive entries allow access, while negative entries deny access.
The authentication succeeds when a matching positive entry is found. The procedure
fails when the first matching negative entry is found, or if no matching entries are
found in either file. The order of entries is important. If the files contain both positive
and negative entries, the entry that appears first will prevail. The rsh(1) and rcp(1)
programs fail if the remote authentication procedure fails. The rlogin program falls
back to the standard password-based login procedure if the remote authentication
fails.

Both files are formatted as a list of one-line entries. Each entry has the form:

hostname [username]

Hostnames must be the official name of the host, not one of its nicknames.

Negative entries are differentiated from positive entries by a ‘−’ character preceding
either the hostname or username field.

Positive Entries
If the form:

hostname

is used, then users from the named host are trusted. That is, they may access the
system with the same user name as they have on the remote system. This form may be
used in both the /etc/hosts.equiv and .rhosts files.

If the line is in the form:

hostname username
then the named user from the named host can access the system. This form may be used in individual .rhosts files to allow remote users to access the system as a different local user. If this form is used in the /etc/hosts.equiv file, the named remote user will be allowed to access the system as any local user.

netgroup(4) can be used in either the hostname or username fields to match a number of hosts or users in one entry. The form:

+@netgroup

allows access from all hosts in the named netgroup. When used in the username field, netgroups allow a group of remote users to access the system as a particular local user. The form:

hostname +@netgroup

allows all of the users in the named netgroup from the named host to access the system as the local user. The form:

+@netgroup1 +@netgroup2

allows the users in netgroup2 from the hosts in netgroup1 to access the system as the local user.

The special character ‘+’ can be used in place of either hostname or username to match any host or user. For example, the entry

+  

will allow a user from any remote host to access the system with the same username. The entry

+ username

will allow the named user from any remote host to access the system. The entry

hostname +

will allow any user from the named host to access the system as the local user.

Negative Entries

Negative entries are preceded by a ‘−’ sign. The form:

−hostname

will disallow all access from the named host. The form:

−@netgroup

means that access is explicitly disallowed from all hosts in the named netgroup. The form:

hostname −username

disallows access by the named user only from the named host, while the form:
will disallow access by all of the users in the named netgroup from all hosts.

**Search Sequence**

To help maintain system security, the `/etc/hosts.equiv` file is not checked when access is being attempted for super-user. If the user attempting access is not the super-user, `/etc/hosts.equiv` is searched for lines of the form described above. Checks are made for lines in this file in the following order:

1. `+`
2. `+@netgroup`
3. `‒@netgroup`
4. `‒hostname`
5. `hostname`

The user is granted access if a positive match occurs. Negative entries apply only to `/etc/hosts.equiv` and may be overridden by subsequent `.rhosts` entries.

If no positive match occurred, the `.rhosts` file is then searched if the user attempting access maintains such a file. This file is searched whether or not the user attempting access is the super-user. As a security feature, the `.rhosts` file must be owned by the user who is attempting access. Checks are made for lines in `.rhosts` in the following order:

1. `+`
2. `+@netgroup`
3. `‒@netgroup`
4. `‒hostname`
5. `hostname`

**FILES**

- `/etc/hosts.equiv`: system trusted hosts and users
- `~/.rhosts`: user’s trusted hosts and users

**SEE ALSO**

rcp(1), rlogin(1), rsh(1), rcmd(3SOCKET), hosts(4), netgroup(4), passwd(4)

**WARNINGS**

Positive entries in `/etc/hosts.equiv` that include a `username` field (either an individual named user, a netgroup, or `+` sign) should be used with extreme caution. Because `/etc/hosts.equiv` applies system-wide, these entries allow one, or a group of, remote users to access the system as any local user. This can be a security hole. For example, because of the search sequence, an `/etc/hosts.equiv` file consisting of the entries

`+`  
`‒@<NAME>`

will not deny access to “hostxxx”.

hosts.equiv(4)
inetd.conf(4)

NAME
inetd.conf – Internet servers database

SYNOPSIS
/etc/inet/inetd.conf

/etc/inetd.conf

DESCRIPTION
The inetd.conf file contains the list of servers that inetd(1M) invokes when it receives an Internet request over a socket. Each server entry is composed of a single line of the form:

```
service-name endpoint-type protocol wait-status uid server-program \ server-arguments
```

Fields are separated by either SPACE or TAB characters. A ‘#’ (number sign) indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines that search this file.

**service-name**
The name of a valid service listed in the services file. For RPC services, the value of the service-name field consists of the RPC service name or program number, followed by a ‘/’ (slash) and either a version number or a range of version numbers (for example, rstatd/2-4).

**endpoint-type**
Can be one of:

- stream for a stream socket
- dgram for a datagram socket
- raw for a raw socket
- seqpacket for a sequenced packet socket
- tli for all TLI endpoints

**protocol**
A recognized protocol listed in the file /etc/inet/protocols. For servers capable of supporting TCP and UDP over IPv6, the following protocol types are also recognized:

```
tcp6
udp6 tcp6 and udp6 are not official protocols; accordingly, they are not listed in the /etc/inet/protocols file.
```

Here the inetd program uses an AF_INET6 type socket endpoint. These servers can also handle incoming IPv4 client requests in addition to IPv6 client requests.

For RPC services, the field consists of the string rpc followed by a ‘/’ (slash) and either a ‘*’ (asterisk), one or more nettypes, one or more netids, or a combination
of nettypes and netids. Whatever the value, it is first treated as a nettype. If it is not a valid nettype, then it is treated as a netid. For example, `rpc/*` for an RPC service using all the transports supported by the system (the list can be found in the `/etc/netconfig` file), equivalent to saying `rpc/visible rpc/tciods` for an RPC service using the Connection-Oriented Transport Service.

**wait-status**

This field has values `wait` or `nowait`. This entry specifies whether the server that is invoked by `inetd` will take over the listening socket associated with the service, and whether once launched, `inetd` will wait for that server to exit, if ever, before it resumes listening for new service requests. The `wait-status` for datagram servers must be set to `wait`, as they are always invoked with the original datagram socket that will participate in delivering the service bound to the specified service. They do not have separate “listening” and “accepting” sockets. Accordingly, do not configure UDP services as `nowait`. This causes a race condition by which the `inetd` program selects on the socket and the server program reads from the socket. Many server programs will be forked, and performance will be severely compromised. Connection-oriented services such as TCP stream services can be designed to be either `wait` or `nowait` status.

**uid**

The user ID under which the server should run. This allows servers to run with access privileges other than those for root.

**server-program**

Either the pathname of a server program to be invoked by `inetd` to perform the requested service, or the value `internal` if `inetd` itself provides the service.

**server-arguments**

If a server must be invoked with command line arguments, the entire command line (including argument 0) must appear in this field (which consists of all remaining words in the entry). If the server expects `inetd` to pass it the address of its peer (for compatibility with 4.2BSD executable daemons), then the first argument to the command should be specified as `'%A'`. No more than 20 arguments are allowed in this field.

**FILES**

- `/etc/netconfig` network configuration file
- `/etc/inet/protocols` Internet protocols

inetd.conf(4)
inetd.conf(4)

/etc/inet/services     Internet network services

SEE ALSO
rlogin(1), rsh(1), in.tftpd(1M), inetd(1M), services(4)

NOTES
/etc/inet/inetd.conf is the official SVR4 name of the inetd.conf file. The
symbolic link /etc/inetd.conf exists for BSD compatibility.
inet_type(4)

NAME  inet_type – default Internet protocol type

SYNOPSIS  /etc/default/inet_type

DESCRIPTION  The inet_type file defines the default IP protocol to use. Currently this file is only used by the ifconfig(1M) and netstat(1M) commands.

The inet_type file can contain a number of <variable>=<value> lines. Currently, the only variable defined is DEFAULT_IP, which can be assigned a value of IP_VERSION4, IP_VERSION6, or BOTH.

The output displayed by the ifconfig and netstat commands can be controlled by the value of DEFAULT_IP set in inet_type file. By default, both commands display the IPv4 and IPv6 information available on the system. The user can choose to suppress display of IPv6 information by setting the value of DEFAULT_IP. The following shows the possible values for DEFAULT_IP and the resulting ifconfig and netstat output that will be displayed:

<table>
<thead>
<tr>
<th>DEFAULT_IP</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP_VERSION4</td>
<td>Displays only IPv4 related information. The output displayed is backward compatible with older versions of the ifconfig(1M) and netstat(1M) commands.</td>
</tr>
<tr>
<td>IP_VERSION6</td>
<td>Displays both IPv4 and IPv6 related information for ifconfig and netstat.</td>
</tr>
<tr>
<td>BOTH</td>
<td>Displays both IPv4 and IPv6 related information for ifconfig and netstat.</td>
</tr>
</tbody>
</table>

The command-line options to the ifconfig and netstat commands override the effect of DEFAULT_IP as set in the inet_type file. For example, even if the value of DEFAULT_IP is IP_VERSION4, the command example% ifconfig -a6 will display all IPv6 interfaces.

EXAMPLES  EXAMPLE 1 Suppressing IPv6 Related Output

This is what the inet_type file must contain if you want to suppress IPv6 related output:

DEFAULT_IP=IP_VERSION4

SEE ALSO  ifconfig(1M), netstat(1M)
init.d – initialization and termination scripts for changing init states

/etc/init.d

 DESCRIPTION
/etc/init.d is a directory containing initialization and termination scripts for changing init states. These scripts are linked when appropriate to files in the rc? .d directories, where ‘?’ is a single character corresponding to the init state. See init(1M) for definitions of the states.

 File names in rc? .d directories are of the form [SK]nn<init.d filename>, where S means start this job, K means kill this job, and nn is the relative sequence number for killing or starting the job. When entering a state (init S,0,2,3,etc.) the rc[S0-6] script executes those scripts in /etc/rc[S0-6].d that are prefixed with K followed by those scripts prefixed with S. When executing each script in one of the /etc/rc[S0-6] directories, the /sbin/rc[S0-6] script passes a single argument. It passes the argument ’stop’ for scripts prefixed with K and the argument ’start’ for scripts prefixed with S. There is no harm in applying the same sequence number to multiple scripts. In this case the order of execution is deterministic but unspecified.

 Guidelines for selecting sequence numbers are provided in README files located in the directory associated with that target state. For example, /etc/rc[S0-6].d/README. Absence of a README file indicates that there are currently no established guidelines.

 EXAMPLES

 EXAMPLE 1 Example of /sbin/rc2.

 When changing to init state 2 (multi-user mode, network resources not exported), /sbin/rc2 is initiated by the init process. The following steps are performed by /sbin/rc2.

 1. In the directory /etc/rc2.d are files used to stop processes that should not be running in state 2. The filenames are prefixed with K. Each K file in the directory is executed (by /sbin/rc2) in alpha-numeric order when the system enters init state 2. See example below.

 2. Also in the rc2.d directory are files used to start processes that should be running in state 2. As in the Step 1, each S file is executed.

 Assume the file /etc/netdaemon is a script that will initiate networking daemons when given the argument ’start’, and will terminate the daemons if given the argument ’stop’. It is linked to /etc/rc2.d/S68netdaemon, and to /etc/rc0.d/K67netdaemon. The file is executed by /etc/rc2.d/S68netdaemon start when init state 2 is entered and by /etc/rc0.d/S67netdaemon stop when shutting the system down.

 SEE ALSO

init(1M)
/sbin/rc2 has references to the obsolescent rc.d directory. These references are for compatibility with old INSTALL scripts. New INSTALL scripts should use the init.d directory for related executables. The same is true for the shutdown.d directory.
NAME

inittab – script for init

DESCRIPTION

The file /etc/inittab controls process dispatching by init. The processes most typically dispatched by init are daemons.

The inittab file is composed of entries that are position dependent and have the following format:

id : rstate : action : process

Each entry is delimited by a newline; however, a backslash (\) preceding a newline indicates a continuation of the entry. Up to 512 characters for each entry are permitted. Comments may be inserted in the process field using the convention for comments described in sh(1). There are no limits (other than maximum entry size) imposed on the number of entries in the inittab file. The entry fields are:

id

One to four characters used to uniquely identify an entry. Do not use the characters "r" or "t" as the first or only character in this field. These characters are reserved for the use of rlogin(1) and telnet(1).

rstate

Define the run level in which this entry is to be processed. Run-levels effectively correspond to a configuration of processes in the system. That is, each process spawned by init is assigned a run level(s) in which it is allowed to exist. The run levels are represented by a number ranging from 0 through 6. For example, if the system is in run level 1, only those entries having a 1 in the rstate field are processed.

When init is requested to change run levels, all processes that do not have an entry in the rstate field for the target run level are sent the warning signal SIGTERM and allowed a 5-second grace period before being forcibly terminated by the kill signal SIGKILL. The rstate field can define multiple run levels for a process by selecting more than one run level in any combination from 0 through 6. If no run level is specified, then the process is assumed to be valid at all run levels 0 through 6.

There are three other values, a, b and c, which can appear in the rstate field, even though they are not true run levels. Entries which have these characters in the rstate field are processed only when an init or telinit process requests them to be run (regardless of the current run level of the system). See init(1M). These differ from run levels in that init can never enter run level a, b or c. Also, a request for the execution of any of these processes does not change the current run level. Furthermore, a process started by an a, b or c command is not killed when init changes levels. They are killed only if their line in inittab is marked off in the action field, their line is deleted entirely from inittab, or init goes into single-user state.
**action**

Key words in this field tell `init` how to treat the process specified in the `process` field. The actions recognized by `init` are as follows:

**resprawm**

If the process does not exist, then start the process; do not wait for its termination (continue scanning the `inittab` file), and when the process dies, restart the process. If the process currently exists, do nothing and continue scanning the `inittab` file.

**wait**

When `init` enters the run level that matches the entry’s `rstate`, start the process and wait for its termination. All subsequent reads of the `inittab` file while `init` is in the same run level cause `init` to ignore this entry.

**once**

When `init` enters a run level that matches the entry’s `rstate`, start the process, do not wait for its termination. When it dies, do not restart the process. If `init` enters a new run level and the process is still running from a previous run level change, the program is not restarted.

**boot**

The entry is to be processed only at `init`'s boot-time read of the `inittab` file. `init` is to start the process and not wait for its termination; when it dies, it does not restart the process. In order for this instruction to be meaningful, the `rstate` should be the default or it must match `init`'s run level at boot time. This action is useful for an initialization function following a hardware reboot of the system.

**bootwait**

The entry is to be processed the first time `init` goes from single-user to multi-user state after the system is booted. (If `initdefault` is set to 2, the process runs right after the boot.) `init` starts the process, waits for its termination and, when it dies, does not restart the process.

**powerfail**

Execute the process associated with this entry only when `init` receives a power fail signal, SIGPWR (see `signal(3C)`).

**powerwait**

Execute the process associated with this entry only when `init` receives a power fail signal, SIGPWR, and wait until it terminates before continuing any processing of `inittab`.

**off**

If the process associated with this entry is currently running, send the warning signal SIGTERM and wait 5 seconds before forcibly terminating the process with the kill signal SIGKILL. If the process is nonexistent, ignore the entry.
ondemand

This instruction is really a synonym for the respawn action. It is functionally identical to respawn but is given a different keyword in order to divorce its association with run levels. This instruction is used only with the a, b or c values described in the rstate field.

initdefault

An entry with this action is scanned only when init is initially invoked. init uses this entry to determine which run level to enter initially. It does this by taking the highest run level specified in the rstate field and using that as its initial state. If the rstate field is empty, this is interpreted as 0123456 and init will enter run level 6. This will cause the system to loop (it will go to firmware and reboot continuously). Additionally, if init does not find an initdefault entry in inittab, it requests an initial run level from the user at reboot time.

sysinit

Entries of this type are executed before init tries to access the console (that is, before the Console Login: prompt). It is expected that this entry will be used only to initialize devices that init might try to ask the run level question. These entries are executed and init waits for their completion before continuing.

process

Specify a command to be executed. The entire process field is prefixed with exec and passed to a forked sh as sh -c 'exec command'. For this reason, any legal sh syntax can appear in the process field.

SEE ALSO

sh(1), who(1), init(1M), ttymon(1M), exec(2), open(2), signal(3C)
The ipnodes file is a local database that associates the names of nodes with their Internet Protocol (IP) addresses. IP addresses can be either an IPv4 or an IPv6 address. The ipnodes file can be used in conjunction with, or instead of, other ipnodes databases, including the Domain Name System (DNS), the NIS ipnodes map, and the NIS+ ipnodes table. Programs use library interfaces to access information in the ipnodes file.

The ipnodes file has one entry for each IP address of each node. If a node has more than one IP address, it will have one entry for each, on consecutive lines. The format of each line is:

```
IP-address official-node-name nicknames...
```

Items are separated by any number of SPACE and/or TAB characters. The first item on a line is the node’s IP address. The second entry is the node’s official name. Subsequent entries on the same line are alternative names for the same machine, or “nicknames.” Nicknames are optional.

For a node with more than one IP address, consecutive entries for these addresses may contain the same or differing nicknames. Different nicknames are useful for assigning distinct names to different addresses.

A call to getipnodebyname(3SOCKET) returns a hostent structure containing the union of all addresses and nicknames from each line containing a matching official name or nickname.

A ‘#’ indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines that search the file.

Network addresses are written in one of two ways:

- The conventional “decimal dot” notation and interpreted using the inet_addr routine from the Internet address manipulation library, inet(3SOCKET).
- The IP Version 6 protocol [IPV6], defined in RFC 1884 and interpreted using the inet_pton() routine from the Internet address manipulation library. See inet(3SOCKET).

These interfaces supports node names as defined in Internet RFC 952 which states:

A "name" (Net, Host, Gateway, or Domain name) is a text string up to 24 characters drawn from the alphabet (A-Z), digits (0-9), minus sign (-), and period (.). Note that periods are only allowed when they serve to delimit components of "domain style names". (See RFC 921, “Domain Name System Implementation Schedule,” for background). No blank or space characters are permitted as part of a name. No distinction is made between upper and lower case. The first character must be an alpha character. The last character must not be a minus sign or period.
Although the interface accepts node names longer than 24 characters for the node portion (exclusive of the domain component), choosing names for nodes that adhere to the 24 character restriction will insure maximum interoperability on the Internet.

A node which serves as a GATEWAY should have "-GATEWAY" or "-GW" as part of its name. Nodes which do not serve as Internet gateways should not use "-GATEWAY" and "-GW" as part of their names. A node that is a TAC should have "-TAC" as the last part of its node name, if it is a DoD node. Single character names or nicknames are not allowed.

RFC 952 has been modified by RFC 1123 to relax the restriction on the first character being a digit.

**EXAMPLE 1** A Typical Line from the ipnodes File

The following is a typical line from the ipnodes file:

```
2::56:a00:20ff:fe7b:b667 foo # John Smith
```

**SEE ALSO**

in.named(1M), getipnodebyname(3SOCKET), inet(3SOCKET), nsswitch.conf(4), resolv.conf(4), hosts(4)


**NOTES**

IPv4 addresses can be defined in the ipnodes file or in the hosts file. See hosts(4). The ipnodes file will be searched for IPv4 addresses when using the getipnodebyname(3SOCKET) API. If no matching IPv4 addresses are found in the ipnodes file, then the hosts file will be searched. To prevent delays in name resolution and to keep /etc/inet/ipnodes and /etc/inet/hosts synchronized, IPv4 addresses defined in the hosts file should be copied to the ipnodes file.
The file /etc/issue contains the issue or project identification to be printed as a login prompt. issue is an ASCII file that is read by program getty and then written to any terminal spawned or respawned from the lines file.

FILES
/etc/issue

SEE ALSO
login(1)
keytables(4)

NAME    keytables – keyboard table descriptions for loadkeys and dumpkeys

DESCRIPTION

These files are used by `loadkeys(1)` to modify the translation tables used by the
keyboard streams module and generated by (see `loadkeys(1)` from those translation
tables.

Any line in the file beginning with `#` is a comment, and is ignored. `#` is treated
specially only at the beginning of a line.

Other lines specify the values to load into the tables for a particular keystation. The
format is either:

```
key number list_of_entries
```
or
```
swap number1 with number2
```
or
```
key number1 same as number2
```
or a blank line, which is ignored.

```
key number list_of_entries
```
sets the entries for keystation `number` from the list given. An entry in that list is of the
form

```
tablename code
```
where `tablename` is the name of a particular translation table, or `all`. The translation
tables are:

```
base                     entry when no shifts are active
shift                    entry when "Shift" key is down
caps                    entry when "Caps Lock" is in effect
ctrl                     entry when "Control" is down
altg                     entry when "Alt Graph" is down
numl                     entry when "Num Lock" is in effect
up                       entry when a key goes up
```

All tables other than `up` refer to the action generated when a key goes down. Entries in
the `up` table are used only for shift keys, since the shift in question goes away when
the key goes up, except for keys such as "Caps Lock" or "Num Lock"; the keyboard
streams module makes the key look as if it were a latching key.
A table name of all indicates that the entry for all tables should be set to the specified value, with the following exception: for entries with a value other than hole, the entry for the numl table should be set to nonl, and the entry for the up table should be set to nop.

The code specifies the effect of the key in question when the specified shift key is down. A code consists of either:

- A character, which indicates that the key should generate the given character. The character can either be a single character, a single character preceded by ^ which refers to a "control character" (for instance, ^c is control-C), or a C-style character constant enclosed in single quote characters (‘), which can be expressed with C-style escape sequences such as \r for RETURN or \000 for the null character. Note that the single character may be any character in an 8-bit character set, such as ISO 8859/1.

- A string, consisting of a list of characters enclosed in double quote characters ("). Note that the use of the double quote character means that a code of double quote must be enclosed in single quotes.

- One of the following expressions:

  - **shiftkeys+leftshift** the key is to be the left-hand "Shift" key
  - **shiftkeys+rightshift** the key is to be the right-hand "Shift" key
  - **shiftkeys+leftctrl** the key is to be the left-hand "Control" key
  - **shiftkeys+rightctrl** the key is to be the right-hand "Control" key
  - **shiftkeys+alt** the key is to be the "Alt" shift key
  - **shiftkeys+altgraph** the key is to be the "Alt Graph" shift key
  - **shiftkeys+capslock** the key is to be the "Caps Lock" key
  - **shiftkeys+shiftlock** the key is to be the "Shift Lock" key
  - **shiftkeys+numlock** the key is to be the "Num Lock" key
  - **buckybits+systembit** the key is to be the "Stop" key in SunView; this is normally the L1 key, or the SETUP key on the VT100 keyboard
  - **buckybits+metabit** the key is to be the "meta" key. That is, the "Left" or "Right" key on a Sun-2 or Sun-3 keyboard or the "diamond" key on a Sun-4 keyboard
  - **compose** the key is to be the "Compose" key
  - **ctrlq** on the "VT100" keyboard, the key is to transmit the control-Q character (this would be the entry for the "Q" key in the ctrl table)
<table>
<thead>
<tr>
<th>key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ctrlS</td>
<td>on the &quot;VT100&quot; keyboard, the key is to transmit the</td>
</tr>
<tr>
<td></td>
<td>control-S character (this would be the entry for the</td>
</tr>
<tr>
<td></td>
<td>&quot;S&quot; key in the ctrl table)</td>
</tr>
<tr>
<td>noscroll</td>
<td>on the &quot;VT100&quot; keyboard, the key is to be the &quot;No</td>
</tr>
<tr>
<td></td>
<td>Scroll&quot; key</td>
</tr>
<tr>
<td>string+uparrow</td>
<td>the key is to be the &quot;up arrow&quot; key</td>
</tr>
<tr>
<td>string+downarrow</td>
<td>the key is to be the &quot;down arrow&quot; key</td>
</tr>
<tr>
<td>string+leftarrow</td>
<td>the key is to be the &quot;left arrow&quot; key</td>
</tr>
<tr>
<td>string+rightarrow</td>
<td>the key is to be the &quot;right arrow&quot; key</td>
</tr>
<tr>
<td>string+homearrow</td>
<td>the key is to be the &quot;home&quot; key</td>
</tr>
<tr>
<td>fa_acute</td>
<td>the key is to be the acute accent &quot;floating accent&quot;</td>
</tr>
<tr>
<td></td>
<td>key</td>
</tr>
<tr>
<td>fa_cedilla</td>
<td>the key is to be the cedilla &quot;floating accent&quot; key</td>
</tr>
<tr>
<td>fa_cflex</td>
<td>the key is to be the circumflex &quot;floating accent&quot; key</td>
</tr>
<tr>
<td>fa_grave</td>
<td>the key is to be the grave accent &quot;floating accent&quot;</td>
</tr>
<tr>
<td></td>
<td>key</td>
</tr>
<tr>
<td>fa_tilde</td>
<td>the key is to be the tilde &quot;floating accent&quot; key</td>
</tr>
<tr>
<td>fa.umlaut</td>
<td>the key is to be the umlaut &quot;floating accent&quot; key</td>
</tr>
<tr>
<td>nonl</td>
<td>this is used only in the Num Lock table; the key is</td>
</tr>
<tr>
<td></td>
<td>not to be affected by the state of Num Lock</td>
</tr>
<tr>
<td>pad0</td>
<td>the key is to be the &quot;0&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>pad1</td>
<td>the key is to be the &quot;1&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>pad2</td>
<td>the key is to be the &quot;2&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>pad3</td>
<td>the key is to be the &quot;3&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>pad4</td>
<td>the key is to be the &quot;4&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>pad5</td>
<td>the key is to be the &quot;5&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>pad6</td>
<td>the key is to be the &quot;6&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>pad7</td>
<td>the key is to be the &quot;7&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>pad8</td>
<td>the key is to be the &quot;8&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>pad9</td>
<td>the key is to be the &quot;9&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>paddot</td>
<td>the key is to be the &quot;.&quot; key on the numeric keypad</td>
</tr>
<tr>
<td>padenter</td>
<td>the key is to be the &quot;Enter&quot; key on the numeric keypad</td>
</tr>
</tbody>
</table>
keytables(4)

| padplus   | the key is to be the "+" key on the numeric keypad |
| padminus  | the key is to be the "−" key on the numeric keypad |
| padstar   | the key is to be the "∗" key on the numeric keypad |
| padslash  | the key is to be the "/" key on the numeric keypad |
| padequal  | the key is to be the "$" key on the numeric keypad |
| padsep    | the key is to be the "," (separator) key on the numeric keypad |
| lf(n)     | the key is to be the left-hand function key n |
| rf(n)     | the key is to be the right-hand function key n |
| tf(n)     | the key is to be the top function key n |
| bf(n)     | the key is to be the "bottom" function key n |
| nop       | the key is to do nothing |
| error     | this code indicates an internal error; to be used only for keystation 126, and must be used there |
| idle      | this code indicates that the keyboard is idle (that is, has no keys down); to be used only for all entries other than the num1 and up table entries for keystation 127, and must be used there |
| oops      | this key exists, but its action is not defined; it has the same effect as nop |
| reset     | this code indicates that the keyboard has just been reset; to be used only for the up table entry for keystation 127, and must be used there. |
| swap number1 with number2 | exchanges the entries for keystations number1 and number2. |
| key number1 same as number2 | sets the entries for keystation number1 to be the same as those for keystation number2. If the file does not specify entries for keystation number2, the entries currently in the translation table are used; if the file does specify entries for keystation number2, those entries are used. |

EXAMPLE 1 Example of setting multiple keystations.

The following entry sets keystation 15 to be a “hole” (that is, an entry indicating that there is no keystation 15); sets keystation 30 to do nothing when Alt Graph is down, generate "!" when Shift is down, and generate "1" under all other circumstances; and sets keystation 76 to be the left-hand Control key.
**EXAMPLE 1** Example of setting multiple keystations.  
(Continued)

key 15  all hole
key 30  base 1 shift ! caps 1 ctrl 1 altg nop
key 76  all shiftkeys+leftctrl up shiftkeys+leftctrl

**EXAMPLE 2** Exchange DELETE and BACKSPACE keys

The following entry exchanges the Delete and Back Space keys on the Type 4 keyboard:

swap 43 with 66

Keystation 43 is normally the Back Space key, and keystation 66 is normally the Delete key.

**EXAMPLE 3** Disable CAPS LOCK key

The following entry disables the Caps Lock key on the Type 3 and U.S. Type 4 keyboards:

key 119 all nop

**EXAMPLE 4** Standard translation tables for the U.S. Type 4 keyboard

The following specifies the standard translation tables for the U.S. Type 4 keyboard:

key 0  all hole
key 1  all buckybits+systembit up buckybits+systembit
key 2  all hole
key 3  all lf(2)
key 4  all hole
key 5  all tf(1)
key 6  all tf(2)
key 7  all tf(10)
key 8  all tf(3)
key 9  all tf(11)
key 10 all tf(4)
key 11 all tf(12)
key 12 all tf(5)
key 13 all shiftkeys+altgraph up shiftkeys+altgraph
key 14 all tf(6)
key 15 all hole
key 16 all tf(7)
key 17 all tf(8)
key 18 all tf(9)
key 19 all shiftkeys+alt up shiftkeys+alt
key 20 all hole
key 21 all rf(1)
key 22 all rf(2)
key 23 all rf(3)
key 24 all hole
key 25 all lf(3)
key 26 all lf(4)
### Example 4

Standard translation tables for the U.S. Type 4 keyboard (Continued)

<table>
<thead>
<tr>
<th>Key</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>all hole</td>
</tr>
<tr>
<td>28</td>
<td>all hole</td>
</tr>
</tbody>
</table>
| 29  | all `\[
| 30  | base 1 shift ` caps 1 ctrl 1 altg nop |
| 31  | base 2 shift @ caps 2 ctrl `@ altg nop |
| 32  | base 3 shift # caps 3 ctrl 3 altg nop |
| 33  | base 4 shift $ caps 4 ctrl 4 altg nop |
| 34  | base 5 shift % caps 5 ctrl 5 altg nop |
| 35  | base 6 shift ^ caps 6 ctrl `^^ altg nop |
| 36  | base 7 shift & caps 7 ctrl `^ altg nop |
| 37  | base 8 shift * caps 8 ctrl `* altg nop |
| 38  | base 9 shift ( caps 9 ctrl 9 altg nop |
| 39  | base 0 shift ) caps 0 ctrl 0 altg nop |
| 40  | base - shift `_ caps 13 ctrl `_ altg nop |
| 41  | base + shift + caps 14 ctrl `^+ altg nop |
| 42  | base ` shift - caps 15 ctrl `^- altg nop |
| 43  | all `\`` |
| 44  | all hole |
| 45  | all rf(4) numl padequal |
| 46  | all rf(5) numl padslash |
| 47  | all rf(6) numl padstar |
| 48  | all bf(13) |
| 49  | all lf(5) |
| 50  | all bf(10) numl padequal |
| 51  | all lf(6) |
| 52  | all hole |
| 53  | all `\` |
| 54  | base q shift Q caps Q ctrl `^Q altg nop |
| 55  | base w shift W caps W ctrl `^W altg nop |
| 56  | base e shift E caps E ctrl `^E altg nop |
| 57  | base r shift R caps R ctrl `^R altg nop |
| 58  | base t shift T caps T ctrl `^T altg nop |
| 59  | base y shift Y caps Y ctrl `^Y altg nop |
| 60  | base u shift U caps U ctrl `^U altg nop |
| 61  | base i shift I caps I ctrl `\` altg nop |
| 62  | base o shift O caps O ctrl `O altg nop |
| 63  | base p shift P caps P ctrl `^P altg nop |
| 64  | base [ shift { caps [ ctrl `^[ altg nop |
| 65  | base ] shift } caps ] ctrl `^] altg nop |
| 66  | all `\`177` |
| 67  | all compose |
| 68  | all rf(7) numl pad7 |
| 69  | all rf(8) numl pad8 |
| 70  | all rf(9) numl pad9 |
| 71  | all bf(15) numl padminus |
| 72  | all lf(7) |
| 73  | all lf(8) |
| 74  | all hole |
| 75  | all hole |
| 76  | all shiftkeys+leftctrl up shiftkeys+leftctrl |
| 77  | base a shift A caps A ctrl `^A altg nop |
| 78  | base s shift S caps S ctrl `^S altg nop |
| 79  | base d shift D caps D ctrl `^D altg nop |
| 80  | base f shift F caps F ctrl `^F altg nop |
EXAMPLE 4 Standard translation tables for the U.S. Type 4 keyboard  (Continued)

<table>
<thead>
<tr>
<th>Key</th>
<th>Base Function</th>
<th>Shift Function</th>
<th>Caps Function</th>
<th>Ctrl Function</th>
<th>Altg Function</th>
<th>Nop Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>81</td>
<td>base g</td>
<td>shift G</td>
<td>caps G</td>
<td>ctrl ^G</td>
<td>altg</td>
<td>nop</td>
</tr>
<tr>
<td>82</td>
<td>base h</td>
<td>shift H</td>
<td>caps H</td>
<td>ctrl <code>\b</code></td>
<td>altg</td>
<td>nop</td>
</tr>
<tr>
<td>83</td>
<td>base j</td>
<td>shift J</td>
<td>caps J</td>
<td>ctrl <code>\n</code></td>
<td>altg</td>
<td>nop</td>
</tr>
<tr>
<td>84</td>
<td>base k</td>
<td>shift K</td>
<td>caps K</td>
<td>ctrl <code>\v</code></td>
<td>altg</td>
<td>nop</td>
</tr>
<tr>
<td>85</td>
<td>base l</td>
<td>shift L</td>
<td>caps L</td>
<td>ctrl <code>L</code></td>
<td>altg</td>
<td>nop</td>
</tr>
<tr>
<td>86</td>
<td>base ;</td>
<td>shift ;</td>
<td>caps ;</td>
<td>ctrl ;</td>
<td>altg</td>
<td>nop</td>
</tr>
<tr>
<td>87</td>
<td>base <code>\\</code></td>
<td>shift <code>&quot;&quot;</code></td>
<td>caps <code>&quot;&quot;</code></td>
<td>ctrl <code>&quot;&quot;</code></td>
<td>altg</td>
<td>nop</td>
</tr>
<tr>
<td>88</td>
<td>base <code>\\</code></td>
<td>shift <code>\</code></td>
<td>caps <code>\</code></td>
<td>ctrl <code>\</code></td>
<td>altg</td>
<td>nop</td>
</tr>
<tr>
<td>89</td>
<td>all <code>\r</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>all bf(11)</td>
<td>numl padenter</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>91</td>
<td>all rf(10)</td>
<td>numl pad4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>92</td>
<td>all rf(11)</td>
<td>numl pad5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>93</td>
<td>all rf(12)</td>
<td>numl pad6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>94</td>
<td>all bf(8)</td>
<td>numl pad0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>95</td>
<td>all lf(9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>96</td>
<td>all hole</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>97</td>
<td>all lf(10)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>98</td>
<td>all shiftkeys+numlock</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>all shiftkeys+leftshift up</td>
<td>shiftkeys+leftshift</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>base z</td>
<td>shift Z</td>
<td>caps Z</td>
<td>ctrl <code>\Z</code></td>
<td>altg</td>
<td>nop</td>
</tr>
<tr>
<td>101</td>
<td>base x</td>
<td>shift X</td>
<td>caps X</td>
<td>ctrl <code>\X</code></td>
<td>altg</td>
<td>nop</td>
</tr>
<tr>
<td>102</td>
<td>base c</td>
<td>shift C</td>
<td>caps C</td>
<td>ctrl <code>\C</code></td>
<td>altg</td>
<td>nop</td>
</tr>
<tr>
<td>103</td>
<td>base v</td>
<td>shift V</td>
<td>caps V</td>
<td>ctrl <code>\V</code></td>
<td>altg</td>
<td>nop</td>
</tr>
<tr>
<td>104</td>
<td>base b</td>
<td>shift B</td>
<td>caps B</td>
<td>ctrl <code>\B</code></td>
<td>altg</td>
<td>nop</td>
</tr>
<tr>
<td>105</td>
<td>base n</td>
<td>shift N</td>
<td>caps N</td>
<td>ctrl <code>\N</code></td>
<td>altg</td>
<td>nop</td>
</tr>
<tr>
<td>106</td>
<td>base m</td>
<td>shift M</td>
<td>caps M</td>
<td>ctrl <code>\M</code></td>
<td>altg</td>
<td>nop</td>
</tr>
<tr>
<td>107</td>
<td>base .</td>
<td>shift &lt;</td>
<td>caps ,</td>
<td>ctrl ,</td>
<td>altg</td>
<td>nop</td>
</tr>
<tr>
<td>108</td>
<td>base .</td>
<td>shift &gt;</td>
<td>caps .</td>
<td>ctrl .</td>
<td>altg</td>
<td>nop</td>
</tr>
<tr>
<td>109</td>
<td>base /</td>
<td>shift ?</td>
<td>caps /</td>
<td>ctrl ^_</td>
<td>altg</td>
<td>nop</td>
</tr>
<tr>
<td>110</td>
<td>all shiftkeys+rightshift up</td>
<td>shiftkeys+rightshift</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>all <code>\n</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>112</td>
<td>all rf(13)</td>
<td>numl pad1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>113</td>
<td>all rf(14)</td>
<td>numl pad2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>114</td>
<td>all rf(15)</td>
<td>numl pad3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>115</td>
<td>all hole</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>116</td>
<td>all hole</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>117</td>
<td>all hole</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>118</td>
<td>all lf(16)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>119</td>
<td>all shiftkeys+capslock</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>all buckybits+metabit up</td>
<td>buckybits+metabit</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>121</td>
<td>base <code> </code> shift <code> </code> caps <code> </code> ctrl <code>@</code> altg <code> </code></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>122</td>
<td>all buckybits+metabit up</td>
<td>buckybits+metabit</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>123</td>
<td>all hole</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>124</td>
<td>all hole</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>125</td>
<td>all bf(14)</td>
<td>numl padplus</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>126</td>
<td>all error</td>
<td>numl error</td>
<td>up hole</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>127</td>
<td>all idle</td>
<td>numl idle</td>
<td>up reset</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SEE ALSO  loadkeys(1)
### NAME
krb5.conf – Kerberos configuration file

### SYNOPSIS
/etc/krb5/krb5.conf

### DESCRIPTION
The krb5.conf file contains Kerberos configuration information, including the locations of KDCs and administration daemons for the Kerberos realms of interest, defaults for the current realm and for Kerberos applications, and mappings of host names onto Kerberos realms. This file must reside on all Kerberos clients.

The format of the krb5.conf consists of sections headings in square brackets. Each section may contain zero or more configuration variables (called relations), of the form:

```
relation = relation-value
```

or

```
relation-subsection = {
relation = relation-value
relation = relation-value
}
```

The krb5.conf file may contain any or all of the following seven sections:

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>libdefaults</td>
<td>Contains default values used by the Kerberos V5 library.</td>
</tr>
<tr>
<td>appdefaults</td>
<td>Contains subsections for Kerberos V5 applications, where relation-subsection is the name of an application. Each subsection describes application-specific defaults.</td>
</tr>
<tr>
<td>realms</td>
<td>Contains subsections for Kerberos realms, where relation-subsection is the name of a realm. Each subsection contains relations that define the properties for that particular realm.</td>
</tr>
<tr>
<td>domain.realm</td>
<td>Contains relations which map domain names and subdomains onto Kerberos realm names. This is used by programs to determine what realm a host should be in, given its fully qualified domain name.</td>
</tr>
<tr>
<td>logging</td>
<td>Contains relations which determine how Kerberos programs are to perform logging.</td>
</tr>
<tr>
<td>capaths</td>
<td>Contains the authentication paths used with direct (nonhierarchical) cross-realm authentication. Entries in this section are used by the client to determine the intermediate realms which may be used in cross-realm authentication.</td>
</tr>
</tbody>
</table>
authentication. It is also used by the end-service when checking the transited field for trusted intermediate realms.

kdc
For a KDC, may contain the location of the kdc.conf file.

[libdefaults]
The [libdefaults] section may contain any of the following relations:

default_realm
Identifies the default Kerberos realm for the client. Set its value to your Kerberos realm.

default_tgs_enctypes
Identifies the supported list of session key encryption types that should be returned by the KDC. The list may be delimited with commas or whitespace. The supported encryption types are des-cbc-crc and des-cbc-md5.

default_tkt_enctypes
Identifies the supported list of session key encryption types that should be requested by the client. The format is the same as for default_tkt_enctypes. The supported encryption types are des-cbc-crc and des-cbc-md5.
clockskew
Sets the maximum allowable amount of clock skew in seconds that the library will tolerate before assuming that a Kerberos message is invalid. The default value is 300 seconds, or five minutes.

[appdefaults]
This section contains subsections for Kerberos V5 applications, where relation-subsection is the name of an application. Each subsection contains relations that define the default behaviors for that application.

gkadmin = {
}

The following application defaults can be set to true or false:

kinit
   forwardable
   proxiable
   renewable
   max_life = delta_time
   max_renewable_life = delta_time

(See kinit(1) for the valid time duration formats you can specify for delta_time.)

In the following example, kinit will get forwardable tickets by default, and telnet has three default behaviors specified:
The application defaults specified here are overridden by those specified in the [realms] section.

This section contains subsections for Kerberos realms, where relation-subsection is the name of a realm. Each subsection contains relations that define the properties for that particular realm. The following relations may be specified in each [realms] subsection:

- **kdc**: The name of a host running a KDC for that realm. An optional port number (separated from the hostname by a colon) may be included.
- **admin_server**: Identifies the host where the Kerberos administration daemon (kadmind) is running. Typically, this is the master KDC.
- **application defaults**: Application defaults that are specific to a particular realm may be specified within a [realms] subsection. Realm-specific application defaults override the global defaults specified in the [appdefaults] section.

This section provides a translation from a domain name or hostname to a Kerberos realm name. The relation can be a host name, or a domain name, where domain names are indicated by a period (‘.’) prefix. relation-value is the Kerberos realm name for that particular host or domain. Host names and domain names should be in lower case.

If no translation entry applies, the host’s realm is considered to be the hostname’s domain portion converted to upper case. For example, the following [domain_realm] section maps crash.mit.edu into the TEST.ATHENA.MIT.EDU realm:

```plaintext
[domain_realm]
.mit.edu = ATHENA.MIT.EDU
mit.edu = ATHENA.MIT.EDU
crash.mit.edu = TEST.ATHENA.MIT.EDU
.fubar.org = FUBAR.ORG
fubar.org = FUBAR.ORG
```
[logging]

This section indicates how Kerberos programs are to perform logging. There are two types of relations for this section: relations to specify how to log and a relation to specify how to rotate kdc log files.

The following relations may be defined to specify how to log. The same relation can be repeated if you want to assign it multiple logging methods.

```
admin_server Specifies how to log the Kerberos administration daemon (kadmind). The default is FILE:/var/krb5/kadmin.log.
default Specifies how to perform logging in the absence of explicit specifications otherwise.
kdc Specifies how the KDC is to perform its logging. The default is FILE:/var/krb5/kdc.log.
```

The admin_server, default, and kdc relations may have the following values:

```
FILE:filename

or

FILE=filename This value causes the entity's logging messages to go to the specified file. If the '=' form is used, the file is overwritten. If the ':' form is used, the file is appended to.

STDERR This value causes the entity's logging messages to go to its standard error stream.

CONSOLE This value causes the entity's logging messages to go to the console, if the system supports it.

DEVICE=devicename This causes the entity's logging messages to go to the specified device.

SYSLOG[:severity [:facility]] This causes the entity's logging messages to go to the system log.
```

The severity argument specifies the default severity of system log messages. This may be any of the following severities supported by the syslog(3C) call, minus the LOG_prefix: LOG_EMERG, LOG_ALERT, LOG_CRIT, LOG_ERR, LOG_WARNING, LOG_NOTICE, LOG_INFO, and LOG_DEBUG. For example, a value of CRIT would specify LOG_CRIT severity.
The facility argument specifies the facility under which the messages are logged. This may be any of the following facilities supported by the syslog(3C) call minus the LOG_prefix: LOG_KERN, LOG_USER, LOG_MAIL, LOG_DAEMON, LOG_AUTH, LOG_LPR, LOG_NEWS, LOG_UUCP, LOG_CRON, and LOG_LOCAL0 through LOG_LOCAL7.

If no severity is specified, the default is ERR. If no facility is specified, the default is AUTH.

The following relation may be defined to specify how to rotate kdc log files if the FILE: value is being used to log:

```
kdc_rotate
```

A relation subsection that enables kdc logging to be rotated to multiple files based on a time interval. This can be used to avoid logging to one file, which may grow too large and bring the KDC to a halt.

The time interval for the rotation is specified by the period relation. The number of log files to be rotated is specified by the versions relation. Both the period and versions (described below) should be included in this subsection. And, this subsection applies only if the kdc relation has a FILE: value.

The following relations may be specified for the kdc_rotate relation subsection:

```
period=delta_time
```

Specifies the time interval before a new log file is created. See the Time Formats section in kinit(1) for the valid time duration formats you can specify for delta_time. If period is not specified or set to "never", no rotation will occur.

Specifying a time interval does not mean that the log files will be rotated at the time interval based on real time. This is because the time interval is checked at each attempt to write a record to the log, or when logging is actually occurring. Therefore, rotation occurs only when logging has actually occurred for the specified time interval.

```
versions=number
```

Specifies how many previous versions will be saved before the rotation begins. A number will be appended to the log file, starting with 0 and ending with (number - 1). For example, if versions is set to 2, up to three logging files will be created (filename, filename.0, and filename.1) before the first one is overwritten to begin the rotation.

Notice that if versions is not specified or set to 0, only one log file will be created, but it will be overwritten whenever the time interval is met.

In the following example, the logging messages from the Kerberos administration daemon will go to the console. The logging messages from the KDC will be appended to the /var/krb5/kdc.log, which will be rotated between twenty-one log files with a specified time interval of a day.
In order to perform direct (non-hierarchical) cross-realm authentication, a database is 
needed to construct the authentication paths between the realms. This section defines 
that database.

A client will use this section to find the authentication path between its realm and the 
realm of the server. The server will use this section to verify the authentication path 
used by the client, by checking the transited field of the received ticket.

There is a subsection for each participating realm, and each subsection has relations 
named for each of the realms. The relation-value is an intermediate realm which may 
participate in the cross-realm authentication. The relations may be repeated if there is 
more than one intermediate realm. A value of '.' means that the two realms share keys 
directly, and no intermediate realms should be allowed to participate.

There are $n^2$ possible entries in this table, but only those entries which will be 
needed on the client or the server need to be present. The client needs a subsection 
named for its local realm, with relations named for all the realms of servers it will 
need to authenticate with. A server needs a subsection named for each realm of the 
clients it will serve.

For example, ANL.GOV, PNL.GOV, and NERSC.GOV all wish to use the ES.NET realm 
as an intermediate realm. ANL has a sub realm of TEST.ANL.GOV, which will 
authenticate with NERSC.GOV but not PNL.GOV. The [capath] section for ANL.GOV 
systems would look like this:

```
[capaths]
ANL.GOV = {
    TEST.ANL.GOV = .
    PNL.GOV = ES.NET
    NERSC.GOV = ES.NET
    ES.NET = .
}
TEST.ANL.GOV = {
    ANL.GOV = .
}
PNL.GOV = {
    ANL.GOV = ES.NET
}
NERSC.GOV = {
    ANL.GOV = ES.NET
}
```
The [capath] section of the configuration file used on NERSC.GOV systems would look like this:

```
[capaths]
NERSC.GOV = {
    ANL.GOV = ES.NET
    TEST.ANL.GOV = ES.NET
    TEST.ANL.GOV = ANL.GOV
    PNL.GOV = ES.NET
    ES.NET = .
}

ANL.GOV = {
    NERSC.GOV = ES.NET
}

PNL.GOV = {
    NERSC.GOV = ES.NET
}

ES.NET = {
    NERSC.GOV = .
}

TEST.ANL.GOV = {
    NERSC.GOV = ANL.GOV
    NERSC.GOV = ES.NET
}
```

In the above examples, the ordering is not important, except when the same relation is used more than once. The client will use this to determine the path. (It is not important to the server, since the transited field is not sorted.)

**EXAMPLE 1** Sample file

Here is an example of a generic krb5.conf file:

```
[libdefaults]
ticket_lifetime = 600
default_realm = ATHENA.MIT.EDU
default_tkt_enctypes = des-cbc-crc
default_tgs_enctypes = des-cbc-crc

[realms]
ATHENA.MIT.EDU = {
    kdc = kerberos.mit.edu
    kdc = kerberos-1.mit.edu
    kdc = kerberos-2.mit.edu
    admin_server = kerberos.mit.edu
}
EXAMPLE 1 Sample file  (Continued)

    default_domain = mit.edu
}

FUBAR.ORG = {
    kdc = kerberos.fubar.org
    kdc = kerberos-1.fubar.org
    admin_server = kerberos.fubar.org
}

[domain_realm]
    .mit.edu = ATHENA.MIT.EDU
    mit.edu = ATHENA.MIT.EDU

FILES /var/krb5/kdc.log  KDC logging file
SEE ALSO  kinit(1), syslog(3C), SEAM(5)

NOTES If the krb5.conf file is not formatted properly, the telnet command will fail. However, the dtlogin and login commands will still succeed, even if the krb5.conf file is specified as required for the commands. If this occurs, the following error message will be displayed:

    Error initializing krb5: Improper format of

To bypass any other problems that may occur, you should fix the file as soon as possible.
**NAME**
ldapfilter.conf – configuration file for LDAP filtering routines

**SYNOPSIS**
/etc/opt/SUNWconn/ldap/current/ldapfilter.conf

**DESCRIPTION**
The ldapfilter.conf file contains information used by the LDAP filtering routines.

Blank lines and lines that begin with a hash character ("#") are treated as comments
and ignored. The configuration information consists of lines that contain one to five
tokens. Tokens are separated by white space, and double quotes can be used to
include white space inside a token.

The file consists of a sequence of one or more filter sets. A filter set begins with a line
containing a single token called a tag.

The filter set consists of a sequence of one or more filter lists. The first line in a filter
list must contain four or five tokens: the value pattern, the delimiter list, a filter template,
a match description, and an optional search scope. The value pattern is a regular
expression that is matched against the value passed to the LDAP library call to select
the filter list.

The delimiter list is a list of the characters (in the form of a single string) that can be
used to break the value into distinct words.

The filter template is used to construct an LDAP filter (see description below)

The match description is returned to the caller along with a filter as a piece of text that
can be used to describe the sort of LDAP search that took place. It should correctly
compete both of the following phrases: "One match description match was found for..."
and "Three match description matches were found for...."

The search scope is optional, and should be one of "base", "onelvel", or "subtree". If
search scope is not provided, the default is "subtree".

The remaining lines of the filter list should contain two or three tokens, a filter template,
a match description and an optional search scope.

The filter template is similar in concept to a printf(3C) style format string. Everything
is taken literally except for the character sequences:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%v</td>
<td>Substitute the entire value string in place of the %v.</td>
</tr>
<tr>
<td>%v$</td>
<td>Substitute the last word in this field.</td>
</tr>
<tr>
<td>%vN</td>
<td>Substitute word N in this field (where N is a single digit 1-9). Words are numbered from left to right within the value starting at 1.</td>
</tr>
<tr>
<td>%vM-N</td>
<td>Substitute the indicated sequence of words where M and N are both single digits 1-9.</td>
</tr>
<tr>
<td>%vN-</td>
<td>Substitute word N through the last word in value where N is again a single digit 1-9.</td>
</tr>
</tbody>
</table>
EXAMPLES

The following ldap filter configuration file contains two filter sets, example1 and example2 onelevel, each of which contains four filter lists.

# ldap filter file
#
example1
"=" " " "%v" "arbitrary filter"

"[0-9][0-9-]*" "=" "(telephoneNumber=*v)" "phone number"

"*" "=" "mail=*v)" "email address"

"^[._]*" "_._" "(cn=%v1 %v2-)*" "first initial"

"^[._]*" "_._" "(cn=%v1-)*" "last initial"

"[._]*" "_._" " (((sn=*v1) (cn=*v1-)) "exact"

"((sn=*v1) (cn=*v1)) "approximate"

"^[._]*" "_._" " (((cn=*v1) (sn=*v1) (uid=*v1)) "exact"

"((cn=*v1) (sn=*v1)) "approximate"

example2 onelevel

"^[._]*" "_._" " (((o=*v) (c=*v) (l=*v) (co=*v)) "exact" "onelevel"

"(((o=*v) (c=*v) (l=*v) (co=*v)) "approximate"

"^[._]*" "_._" " (((o=*v) (c=*v) (l=*v) (co=*v)) "exact"

"((o=*v) (c=*v) (l=*v) (co=*v)) "approximate"

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>SUNWlDap (32-bit)</td>
</tr>
<tr>
<td></td>
<td>SUNWlDapx (64-bit)</td>
</tr>
<tr>
<td>Stability Level</td>
<td>Evolving</td>
</tr>
</tbody>
</table>

SEE ALSO

ldap_getfilter(3LDAP), ldap_ufn(3LDAP), attributes(5)
### NAME
ldapsearchprefs.conf – configuration file for LDAP search preference routines

### SYNOPSIS
/etc/opt/SUNWconn/ldap/current/ldapsearchprefs.conf

### DESCRIPTION
The ldapsearchprefs.conf file contains information used by LDAP when searching the directory. Blank lines and lines that start with a hash (#) character are treated as comments and ignored. Non-comment lines contain one or more tokens. Tokens are separated by white space, and double quotes can be used to include white space inside a token.

Search preferences are typically used by LDAP-based client programs to specify what a user may search for, which attributes are searched, and which options are available to the user.

The first non-comment line specifies the version of the template information and must contain the token Version followed by an integer version number. For example:

Version 1
The current version is 1, so the above example is always the correct opening line.

The remainder of the file consists of one or more search preference configurations. The first line of a search preference is a human-readable name for the type of object being searched for, for example People or Organizations. This name is stored in the so_objtypetypemove of the ldap_searchobj structure (see ldap_searchprefs(3LDAP)). For example,

People
specifies a label for a search preference designed to find X.500 entries for people.

The next line specifies a list of options for this search object. The only option currently allowed is “internal” which means that this search object should not be presented directly to a user. Options are placed in the so_options member of the ldap_searchobj structure and can be tested using the LDAP_IS_SEARCHOBJ_OPTION_SET() macro. Use “” if no special options are required.

The next line specifies a label to use for “Fewer Choices” searches. “Fewer Choices” searches are those where the user’s input is fed to the ldap_filter routines to determine an appropriate filter to use. This contrasts with explicitly-constructed LDAP filters, or “More Choices” searches, where the user can explicitly construct an LDAP filter.

For example:

"Search For:" can be used by LDAP client programs to label the field into which the user can type a "Fewer Choices" search.

The next line specifies an LDAP filter prefix to append to all "More Choices" searched. This is typically used to limit the types of entries returned to those containing a specific object class. For example:
"(&objectClass=person)*" would cause only entries containing the object class person to be returned by a search. Note that parentheses may be unbalanced here, since this is a filter prefix, not an entire filter.

The next line is an LDAP filter tag which specifies the set of LDAP filters to be applied for "Fewer Choices" searching. The line

"x500-People"

would tell the client program to use the set of LDAP filters from the ldap filter configuration file tagged "x500-People".

The next line specifies an LDAP attribute to retrieve to help the user choose when several entries match the search terms specified. For example:

"title"
specifies that if more than one entry matches the search criteria, the client program should retrieve the title attribute that and present that to the user to allow them to select the appropriate entry. The next line specifies a label for the above attribute, for example,

"Title:"

Note that the values defined so far in the file are defaults, and are intended to be overridden by the specific search options that follow.

The next line specifies the scope of the LDAP search to be performed. Acceptable values are subtree, onelevel, and base.

The next section is a list of "More Choices" search options, terminated by a line containing only the string END. For example:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Name</td>
<td>cn</td>
</tr>
<tr>
<td>Surname</td>
<td>sn</td>
</tr>
<tr>
<td>Business Phone</td>
<td>telephoneNumber</td>
</tr>
</tbody>
</table>

Each line represents one method of searching. In this example, there are three ways of searching - by Common Name, by Surname, and by Business Phone number. The first field is the text which should be displayed to user. The second field is the attribute which will be searched. The third field is a bitmap which specifies which of the match types are permitted for this search type. A "1" value in a given bit position indicates that a particular match type is valid, and a "0" indicates that is it not valid. The fourth and fifth fields are, respectively, the select attribute name and on-screen name for the selected attribute. These values are intended to override the defaults defined above. If no specific values are specified, the client software uses the default values above.

The next section is a list of search match options, terminated by a a line containing only the string END. Example:

<table>
<thead>
<tr>
<th>Match Type</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>exactly matches</td>
<td>&quot;($a=$v)&quot;</td>
</tr>
<tr>
<td>approximately matches</td>
<td>&quot;($a~=$v)&quot;</td>
</tr>
<tr>
<td>starts with</td>
<td>&quot;($a=$v*)&quot;</td>
</tr>
<tr>
<td>ends with</td>
<td>&quot;($a=*$v)&quot;</td>
</tr>
<tr>
<td>contains</td>
<td>&quot;($a=<em>$v</em>)&quot; END</td>
</tr>
</tbody>
</table>
In this example, there are five ways of refining the search. For each method, there is an LDAP filter suffix which is appended to the ldap filter.

EXAMPLE 1 The following example illustrates one possible configuration of search preferences for "people".

# Version number
Version 1
# Name for this search object
People
# Label to place before text box user types in
"Search For:"
# Filter prefix to append to all "More Choices" searches
"(&(objectClass=person)"
# Tag to use for "Fewer Choices" searches - from ldapfilter.conf file
"x500-People"
# If a search results in > 1 match, retrieve this attribute to help
# user distinguish between the entries...
multilineDescription
# ...and label it with this string:
"Description"
# Search scope to use when searching
subtree
# Follows a list of "More Choices" search options. Format is:
# Label, attribute, select-bitmap, extra attr display name, extra attr ldap name
# If last two are null, "Fewer Choices" name/attributes used
"Common Name" cn 11111 "" ""
"Surname" sn 11111 "" ""
"Business Phone" "telephoneNumber" 11101 "" ""
"E-Mail Address" "mail" 11111 "" ""
"Uniname" "uid" 11111 "" ""
END
# Match types
"exactly matches" "(%a=%v)"
"approximately matches" "(%a~=%v)"
"starts with" "(%a=%v*)"
"ends with" "(%a=*%v)"
"contains" "(%a=*%v*)"
END

In this example, the user may search for People. For "fewer choices" searching, the tag for the ldapfilter.conf(4) file is "x500-People".

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>SUNWldap (32-bit)</td>
</tr>
<tr>
<td></td>
<td>SUNWldapx (64-bit)</td>
</tr>
<tr>
<td>Stability Level</td>
<td>Evolving</td>
</tr>
</tbody>
</table>

SEE ALSO ldap_searchprefs(3LDAP) attributes(5)
ldaptemplates.conf – configuration file for LDAP display template routines

/etc/opt/SUNWconn/ldap/current/ldaptemplates.conf

The ldaptemplates.conf file contains information used by the LDAP display routines.

Blank lines and lines that start with a hash character ("#") are treated as comments and ignored. Non-comment lines contain one or more tokens. Tokens are separated by white space, and double quotes can be used to include white space inside a token.

The first non-comment line specifies the version of the template information and must contain the token Version followed by an integer version number. For example,

```
Version 1
```

The current version is 1, so the above example is always the correct first line.

The remainder of the file consists of one or more display templates. The first two lines of the display template each contain a single token that specifies singular and plural names for the template in a user-friendly format. For example,

```
"Person"
"People"
```
specifies appropriate names for a template designed to display person information.

The next line specifies the name of the icon or similar element that is associated with this template. For example,

```
"person icon"
```

The next line is a blank-separated list of template options. "" can be used if no options are desired. Available options are: addable (it is appropriate to allow entries of this type to be added), modrdn (it is appropriate to offer the modify rdn operation), altview (this template is an alternate view of another template). For example,

```
"addable" "modrdn"
```

The next portion of the template is a list of X.500 object classes that is used to determine whether the template should be used to display a given entry. The object class information consists of one or more lines, followed by a terminating line that contains the single token END. Each line contains one or more object class names, all of which must be present in a directory entry. Multiple lines can be used to associate more than one set of object classes with a given template. For example,

```
emailPerson
orgPerson
END
```
means that the template is appropriate for display of emailPerson entries or orgPerson entries.

The next line after the object class list is the name of the attribute to authenticate as to make changes (use "" if it is appropriate to authenticate as the entry itself). For example,
"owner"

The next line is the default attribute to use when naming a new entry, for example, "cn"

The next line is the distinguished name of the default location under which new entries are created. For example, "o=XYZ, c=US"

The next section is a list of rules used to assign default values to new entries. The list should be terminated with a line that contains the single token END. Each line in this section should either begin with the token constant and be followed by the name of the attribute and a constant value to assign, or the line should begin with addersdn followed by the name of an attribute whose value will be the DN of the person who has authenticated to add the entry. For example,

constant  associatedDomain  XYZ.us
addersdn    seeAlso
END

The last portion of the template is a list of items to display. It consists of one or more lines, followed by a terminating line that contains the single token END. Each line must begin with the token samerow or the token item

It is assumed that each item appears on a row by itself unless it was preceded by a samerow line (in which case it should be displayed on the same line as the previous item, if possible). Lines that begin with samerow should not have any other tokens on them.

Lines that begin with item must have at least three more tokens on them: an item type, a label, and an attribute name. Any extra tokens are taken as extra arguments.

The item type token must be one of the following strings:
cis case-ignore string attributes
mls multiline string attributes
mail RFC-822 conformant mail address attributes
dn distinguished name pointer attributes
bool Boolean attributes
jpeg JPEG photo attributes
jpegbtn a button that will retrieve and show a JPEG photo attribute
fax FAX T4 format image attributes
faxbtn a button that will retrieve and show a FAX photo attribute
audiobtn audio attributes
time UTC time attributes
date UTC time attributes where only the date portion should be shown
url labeled Uniform Resource Locator attributes
searchact define an action that will do a directory search for other entries
linkact define an action which is a link to another display template
protected for an encrypted attribute, with values displayed as asterisks

An example of an item line for the drink attribute (displayed with label "Work Phone"):

```
item cis "Work Phone" telephoneNumber
```

### EXAMPLES

#### EXAMPLE 1

The following template configuration file contains a templates for display of people entries.

```
# LDAP display templates
#
# Version must be 1 for now
#
Version 1
#
# Person template
"Person"
"People"

# name of the icon that is associated with this template
"person icon"

# blank-separated list of template options ("" for none)
"addable"

#
# objectclass list
person
END

#
# name of attribute to authenticate as ("" means auth as this entry)
"

#
# default attribute name to use when forming RDN of a new entry
#
"cn"

#
# default location when adding new entries (DN; "" means no default)
"o-XYZ, c-US"

#
# rules used to define default values for new entries
END
```
EXAMPLE 1 The following template configuration file contains a template for display of people entries.  (Continued)

```
#
# list of items for display
item jpegbtn  "View Photo"  jpegPhoto  "Next Photo"
item audiobtn  "Play Sound"  audio
item cis  "Also Known As"  cn
item cis  "Title"  title
item mls  "Work Address"  postalAddress
item cis  "Work Phone"  telephoneNumber
item cis  "Fax Number"  facsimileTelephoneNumber
item mls  "Home Address"  homePostalAddress
item cis  "Home Phone"  homePhone
item cis  "User ID"  uid
item mail  "E-Mail Address"  mail
item cis  "Description"  description
item dn  "See Also"  seeAlso
END
```

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>SUNWldap (32-bit)</td>
</tr>
<tr>
<td>Stability Level</td>
<td>Evolving</td>
</tr>
</tbody>
</table>

SEE ALSO  ldap_disptmpl(3LDAP)  ldap_entry2text(3LDAP)  attributes(5)
NAME       limits – header for implementation-specific constants
SYNOPSIS   #include <limits.h>
DESCRIPTION The header <limits.h> is a list of minimal magnitude limitations imposed by a
specific implementation of the operating system.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_ARG_MAX32</td>
<td>1048320</td>
<td>/* max length of arguments to exec 32-bit program */</td>
</tr>
<tr>
<td>_ARG_MAX64</td>
<td>2096640</td>
<td>/* max length of arguments to exec 64-bit program */</td>
</tr>
<tr>
<td>CHAR_BIT</td>
<td>8</td>
<td>/* max # of bits in a char */</td>
</tr>
<tr>
<td>CHAR_MAX</td>
<td>255</td>
<td>/* max value of a char */</td>
</tr>
<tr>
<td>CHAR_MIN</td>
<td>0</td>
<td>/* min value of a char */</td>
</tr>
<tr>
<td>CHILD_MAX</td>
<td>25</td>
<td>/* max # of processes per user id */</td>
</tr>
<tr>
<td>CLK_TCK</td>
<td>_sysconf(3)</td>
<td>/* clock ticks per second */</td>
</tr>
<tr>
<td>DBL_DIG</td>
<td>15</td>
<td>/* digits of precision of a double */</td>
</tr>
<tr>
<td>DBL_MAX</td>
<td>1.7976931348623157E+308</td>
<td>/* max decimal value of a double */</td>
</tr>
<tr>
<td>DBL_MIN</td>
<td>2.2250738585072014E-308</td>
<td>/* min decimal value of a double */</td>
</tr>
<tr>
<td>FCHR_MAX</td>
<td>1048576</td>
<td>/* historical default file size limit in bytes */</td>
</tr>
<tr>
<td>FLT_DIG</td>
<td>6</td>
<td>/* digits of precision of a float */</td>
</tr>
<tr>
<td>FLT_MAX</td>
<td>3.40282347e+38F</td>
<td>/* max decimal value of a float */</td>
</tr>
<tr>
<td>FLT_MIN</td>
<td>1.17549435E-38F</td>
<td>/* min decimal value of a float */</td>
</tr>
<tr>
<td>INT_MAX</td>
<td>2147483647</td>
<td>/* max value of an int */</td>
</tr>
<tr>
<td>INT_MIN</td>
<td>(-2147483647-1)</td>
<td>/* min value of an int */</td>
</tr>
<tr>
<td>LINK_MAX</td>
<td>1000</td>
<td>/* max # of links to a single file */</td>
</tr>
<tr>
<td>LOGNAME_MAX</td>
<td>8</td>
<td>/* max # of characters in a login name */</td>
</tr>
<tr>
<td>LONG_BIT</td>
<td>32</td>
<td>/* # of bits in a long */</td>
</tr>
<tr>
<td>LONG_MAX</td>
<td>2147483647L</td>
<td>/* max value of a long int if _ILP32 defined */</td>
</tr>
<tr>
<td></td>
<td>9223372036854775807L</td>
<td>/* max value of a long int if _LP64 defined */</td>
</tr>
<tr>
<td>Variable</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LONG_MIN</td>
<td>(-2147483647L)</td>
<td>/* min value of a long int if _ILP32 defined */</td>
</tr>
<tr>
<td></td>
<td>(-9223372036854775807L-1L)</td>
<td>/* min value of a long int if _LP64 defined */</td>
</tr>
<tr>
<td>MAX_CANON</td>
<td>256</td>
<td>/* max bytes in a line for canonical processing */</td>
</tr>
<tr>
<td>MAX_INPUT</td>
<td>512</td>
<td>/* max size of a char input buffer */</td>
</tr>
<tr>
<td>MB_LEN_MAX</td>
<td>5</td>
<td>/* max # of bytes in a multibyte character */</td>
</tr>
<tr>
<td>NAME_MAX</td>
<td>14</td>
<td>/* max # of characters in a file name */</td>
</tr>
<tr>
<td>NGROUPS_MAX</td>
<td>16</td>
<td>/* max # of groups for a user */</td>
</tr>
<tr>
<td>NL_ARGMAX</td>
<td>9</td>
<td>/* max value of 'digit' in calls to the NLS printf() and scanf() */</td>
</tr>
<tr>
<td>NL_LANGMAX</td>
<td>14</td>
<td>/* max # of bytes in a LANG name */</td>
</tr>
<tr>
<td>NL_MSGMAX</td>
<td>32767</td>
<td>/* max message number */</td>
</tr>
<tr>
<td>NL_NMAX</td>
<td>1</td>
<td>/* max # of bytes in N-to-1 mapping characters */</td>
</tr>
<tr>
<td>NL_SETMAX</td>
<td>255</td>
<td>/* max set number */</td>
</tr>
<tr>
<td>NL_TEXTMAX</td>
<td>255</td>
<td>/* max # of bytes in a message string */</td>
</tr>
<tr>
<td>NZERO</td>
<td>20</td>
<td>/* default process priority */</td>
</tr>
<tr>
<td>OPEN_MAX</td>
<td>20</td>
<td>/* max # of files a process can have open */</td>
</tr>
<tr>
<td>PASS_MAX</td>
<td>8</td>
<td>/* max # of characters in a password */</td>
</tr>
<tr>
<td>PATH_MAX</td>
<td>1024</td>
<td>/* max # of characters in a path name */</td>
</tr>
<tr>
<td>PID_MAX</td>
<td>999999</td>
<td>/* max value for a process ID */</td>
</tr>
<tr>
<td>PIPE_BUF</td>
<td>5120</td>
<td>/* max # bytes atomic in write to a pipe */</td>
</tr>
<tr>
<td>PIPE_MAX</td>
<td>5120</td>
<td>/* max # bytes written to a pipe in a write */</td>
</tr>
<tr>
<td>SCHAR_MAX</td>
<td>127</td>
<td>/* max value of a 'signed char' */</td>
</tr>
<tr>
<td>SCHAR_MIN</td>
<td>(-128)</td>
<td>/* min value of a 'signed char' */</td>
</tr>
<tr>
<td>SHIRT_MAX</td>
<td>32767</td>
<td>/* max value of a 'short int' */</td>
</tr>
</tbody>
</table>
SHRT_MIN (-32768) /* min value of a "short int" */
STD_BLK 1024 /* # bytes in a physical I/O block */
SYS_NMLN 257 /* 4.0 size of utxname elements */
/* also defined in sys/utsname.h */
SYSPID_MAX 1 /* max pid of system processes */
TMP_MAX 17576 /* max # of unique names generated by tmpnam */
UCHAR_MAX 255 /* max value of an "unsigned char" */
UID_MAX 2147483647 /* max value for a user or group ID */
UINT_MAX 4294967295 /* max value of an "unsigned int" */
ULONG_MAX 4294967295UL /* max value of an "unsigned long int" */
18446744073709551615UL /* max value of an "unsigned long int" */
if _ILP32 defined */
USHRT_MAX 65535 /* max value of an "unsigned short int" */
USI_MAX 4294967295 /* max decimal value of an "unsigned" */
WORD_BIT 32 /* # of bits in a word or int */

The following POSIX definitions are the most restrictive values to be used by a
POSIX-conforming application (see standards(5)). Conforming implementations
shall provide values at least this large.

_POSIX_ARG_MAX 4096 /* max length of arguments to exec */
_POSIX_CHILD_MAX 6 /* max # of processes per user ID */
_POSIX_LINK_MAX 8 /* max # of links to a single file */
_POSIX_MAX_CANON 255 /* max # of bytes in a line of input */
_POSIX_MAX_INPUT 255 /* max # of bytes in terminal input queue */
_POSIX_NAME_MAX 14 /* # of bytes in a filename */
_POSIX_NGROUPS_MAX 0 /* max # of groups in a process */
_POSIX_OPEN_MAX 16 /* max # of files a process can have open */
_POSIX_PATH_MAX 255 /* max # of characters in a pathname */
SEE ALSO

 standards(5)
NAME  llc2 – LLC2 Configuration file

SYNOPSIS  /etc/llc2/default/llc2.*

DESCRIPTION  The llc2 files contain information needed by LLC2 to establish the appropriate links to the underlying MAC layer drivers as well as the parameters necessary to configure the LLC (Logical Link Control) Class II Station Component structures for that link.

The comments are made up of one or more lines starting with the "#" character in column 1.

The main section consists of keyword/value pairs of the form keyword=value, used to initialize the particular adapter.

A sample of the llc2 is presented below:

devicename=/dev/dnet
deviceinstance=1
llc2_on=1  # LLC2: On/Off on this device
deviceloopback=1
timeinterval=0  # LLC2: Timer Multiplier
acktimer=2    # LLC2: Ack Timer
reptimer=2    # LLC2: Response Timer
polltimer=4   # LLC2: Poll Timer
rejecttimer=6 # LLC2: Reject Timer
rembusytimer=8 # LLC2: Remote Busy Timer
inacttimer=30 # LLC2: Inactivity Timer
maxretry=6    # LLC2: Maximum Retry Value
xmitwindowsz=14 # LLC2: Transmit Window Size
rcvwindowsz=14 # LLC2: Receive Window Size

MAC specific Parameters

The llc2.ppa file contains 4 parameters directly related to the underlying MAC-level driver. These are the name of the physical device, the instance of the device, whether LLC2 can be used with this device, and whether the device is capable of looping back data addressed to the node’s unique MAC address, broadcast address, or multicast addresses.

Setting the llc2_on parameter to 1 means that LLC2 can be used with this device; setting it to 0 means otherwise. Setting the loopback parameter to 1 means that the LLC2 module will loop back data addressed to this node’s unique MAC address or to a broadcast/multicast address.

The most likely use is for a media that cannot receive its own transmissions (for example, ethernet) or when the MAC-level driver intentionally does not loop back data addressed to the local node under the assumption that the upper layers have already done so.

Host-Based LLC2 Parameters

The LLC2 contains ten parameters in the configuration file (/etc/llc2/default/llc2.ppa) that apply to configurations using the Host-Based LLC2 component for connection-oriented operation over an Ethernet, Token Ring, or FDDI media.

The ten parameters break down into the following four groups:
Six parameters deal with timer settings for managing the flow of LLC elements of procedure (PDUs) on a data link connection.

One parameter is the multiplier that is used to determine the period of the interval timer for the station. A value of 1 means that each tick count represents 100 milliseconds; 5 means each tick count is 500 milliseconds. Should the parameter be omitted, the default value is 5, except for Token Ring links which use a default of 1.

One parameter indicates how many times an operation should be retried on a data link connection.

Two parameters are for controlling the number of unacknowledged I PDUs to send or receive on a data link connection.

Additional information on these parameters can be found in ISO 8802-2: 1989, Section 7.8.

The following table of Logical Link Control Parameters provides the LLC configuration parameter names, default values, and ranges.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>timeinterval</td>
<td>The timer ticks in 100 ms intervals. This parameter is used to scale the following 5 timer parameters.</td>
<td>5, except TPR: 1</td>
<td>0-10</td>
</tr>
<tr>
<td>acktimer</td>
<td>The connection acknowledgment timer length in (100 * timeinterval) ms.</td>
<td>2</td>
<td>&gt;0</td>
</tr>
<tr>
<td>rsptimer</td>
<td>The response acknowledgment timer length in (100 * timeinterval) ms.</td>
<td>2</td>
<td>&gt;0</td>
</tr>
<tr>
<td>polltimer</td>
<td>The connection poll timer length in (100 * timeinterval) ms.</td>
<td>4</td>
<td>&gt;0</td>
</tr>
<tr>
<td>rejecttimer</td>
<td>The connection reject timer length in (100 * timeinterval) ms.</td>
<td>6</td>
<td>&gt;0</td>
</tr>
<tr>
<td>rembusytimer</td>
<td>The connection remote busy timer length in (100 * timeinterval) ms.</td>
<td>8</td>
<td>&gt;0</td>
</tr>
</tbody>
</table>
### llc2(4)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>inacttimer</td>
<td>The connection inactivity timer length in (100 * timeinterval) ms.</td>
<td>30</td>
<td>&gt; 0</td>
</tr>
<tr>
<td>maxretry</td>
<td>The maximum number of retries of an action on a connection.</td>
<td>6</td>
<td>0 - 100</td>
</tr>
<tr>
<td>xmitwindowsz</td>
<td>The maximum number of unacknowledged I-format protocol data units that can be transmitted on a connection before awaiting an acknowledgment.</td>
<td>14</td>
<td>0 - 127</td>
</tr>
<tr>
<td>rcvwindowsz</td>
<td>The maximum number of unacknowledged I-format protocol data units that can be received on a connection before an acknowledgment is sent.</td>
<td>14</td>
<td>0 - 127</td>
</tr>
</tbody>
</table>

Default values are set when the following conditions are true:
- The parameter is not set by the user.
- The user requests a default `/etc/ildcf` file built based on the adapters installed.
- The user codes a value of 0 for a parameter.

**Timer Parameter Descriptions**

The `acktimer` parameter is used to manage the following sample sequences:

1. Attempting to establish, reset, or disconnect a connection.

   - SABME start acknowledgment timer
   - DISC

   The acknowledgment timer expires before the receipt of a response.

   - SABME start acknowledgment timer
   - DISC

   stop acknowledgment timer
   <-------------------------- UA

2. Sending an FRMR in response to a received PDU of dubious distinction:

   - PDU with invalid N(R)
   - or
   - I PDU with invalid N(S)
   - or
   - PDU of invalid length
   - or
unexpected UA PDU or response PDU with invalid P/F setting

start acknowledgment timer
FRMR ------------------------>

Acknowledgment timer expires before the receipt of a PDU.

start acknowledgment timer
FRMR ------------------------>

stop acknowledgment timer
SABME, FRMR
<------------------------ DISC, or DM

3. There is also a special case of the acknowledgment timer, referred to in this implementation as the response acknowledgment timer (rsptimer). It is used when sending an I PDU.

start response acknowledgement timer
I ------------------------>

Response acknowledgment timer expires before the receipt of an acknowledgment.

start poll timer
RR ------------------------>

polltimer

The polltimer parameter is used to manage situations where a Supervisory command PDU (RR, RNR, or REJ) is sent with the P/F bit set. This type of PDU is typically sent when:

- There has been a period of inactivity on a connection in information transfer mode.
- The remote node must be notified of a local busy condition occurring in information transfer mode.

The expiration of the poll timer causes another Supervisory command PDU (which may be of a different type than the first) to be sent with the P/F bit set, provided the retry count has not exceeded the maximum retry value. This timer, then, provides an extended retry mechanism for a connection in information transfer mode.

rejcttimer

The rejecttimer parameter controls the frequency with which a REJ PDU is sent to a remote node from which an I PDU with an unexpected N(S) was received and which has not corrected the situation by sending an I PDU with the expected N(S).

<------------------------ I PDU with unexpected N(S)

start reject timer
Reject timer expires before the receipt of an I PDU with an expected N(S).

- **start reject and poll timer**
- **stop reject and poll timer**

**rembusytimer**
The rembusytimer parameter is used to determine how long the local node should wait, after the remote node sends an RNR to indicate it is busy, before sending a Supervisory PDU with the P/F bit set to solicit the current state of the remote node. If the remote node indicates that it has cleared its busy condition before the timer expires, the local node stops the remote busy timer.

**inacttimer**
The inacttimer parameter controls how much time is allowed to elapse on a connection in information transfer mode between the issuing of command PDUs by the local node. If the inactivity timer expires because a command PDU has not been generated in the configured time interval, a Supervisory PDU with the P/F bit set is sent to the remote node to solicit its current state, provided that the connection is in information transfer mode. Each time a command PDU is sent by the local node, the inactivity timer is restarted.

The following rules of thumb should apply for the timer parameters:

- The **acktimer**, **rsptimer**, and **polltimer** parameters should have small relative values to allow for quick recovery from common transient error conditions on a connection.
- The **rejecttimer** and **rembusytimer** parameters should have intermediate relative values to allow the local and remote nodes time to recover without resorting to possibly unnecessary polling cycles.
- The **inacttimer** parameter should be set to a large relative value to provide a safety net in information transfer mode.

You may need to shift the values for the timer parameters to higher values if bridges are included in the network or a user application requires a substantial amount of time to respond to connection establishment requests or handle information flow.

The **maxretry** parameter determines the number of times a recovery operation is performed before notifying the user that an error has occurred on a connection. Typical examples of its use include the following:

- When the remote node fails to respond to a SABME sent by the local node to establish or reset the connection, the SABME is resent each time the acknowledgment timer expires, up to maxretry number of times.
In information transfer mode, if the response acknowledgment timer expires after an I PDU has been sent, an RR with the P/F bit set is sent (and resent each time the poll timer expires) until the remote node responds or maxretry number of RRs have been sent.

In general, the maxretry value should not need to be large. Since the acknowledgment and poll timers are typically used in recovery operations that involve the maxretry parameter, the product of maxretry and either acktimer, rsptimer, or polltimer gives a rough estimate of the length of time allotted for the connection to attempt internal error recovery before notifying the user.

rcvwindowsz The rcvwindowsz parameter is used to set the receive window size for I PDUs received locally on a connection. This value should agree with the transmit window size set for the connection at the remote node. If the local rcvwindowsz is greater than the remote transmit window size, I PDUs sent by the remote node are not acknowledged quickly. If the local rcvwindowsz is less than the remote transmit window size, there is a greater risk of the local node generating FRMR PDUs, requiring intervention by the user application when transient errors on the connection require the remote node to retransmit an I PDU. REJ PDUs are recovered internally.

xmitwindowsz The xmitwindowsz parameter sets the local transmit window size for a connection. It denotes the number of unacknowledged I PDUs that the local node may have outstanding. The configured value should match the receive window size for the connection at the remote node, based on the same reasoning as for the rcvwindowsz parameter.

In many cases, the values assigned to rcvwindowsz and xmitwindowsz for adapters on a server node will depend on the transmit and receive window sizes specified for another LLC implementation on a client node. In cases where this LLC implementation is resident in both nodes, larger values for these parameters are useful in environments where much of the activity on a connection consists of file transfer operations. Smaller values are warranted if analysis of LLC2 connection component statistics reveals that connections are entering local or remote busy state frequently.

For a complete explanation of the keywords used, see the publication, The Logical Link Control Driver for Solaris, Installation and Diagnostics.

FILES /etc/llc2/default/llc2.*
SEE ALSO llc2_autoconfig(1), llc2_config(1), llc2(7D)
The /etc/logindevperm file contains information that is used by login(1) and ttymon(1M) to change the owner, group, and permissions of devices upon logging into or out of a console device. By default, this file contains lines for the keyboard, mouse, audio, and frame buffer devices.

The owner of the devices listed in /etc/logindevperm is set to the owner of the console by login(1). The group of the devices is set to the owner's group specified in /etc/passwd. The permissions are set as specified in /etc/logindevperm.

Fields are separated by TAB and/or SPACE characters. Blank lines and comments can appear anywhere in the file; comments start with a hashmark, '#', and continue to the end of the line.

The first field specifies the name of a console device (for example, /dev/console). The second field specifies the permissions to which the devices in the device_list field (third field) will be set. A device_list is a colon-separated list of device names. A device entry that is a directory name and ends with "/*" specifies all entries in the directory (except "." and ".".). For example, "/dev/fbs/*" specifies all frame buffer devices.

Once the devices are owned by the user, their permissions and ownership can be changed using chmod(1) and chown(1), as with any other user-owned file.

Upon logout the owner and group of these devices will be reset by ttymon(1M) to owner root and root's group as specified in /etc/passwd (typically other). The permissions are set as specified in the /etc/logindevperm file.

See also chmod(1), chown(1), login(1), ttymon(1M), passwd(4)

Notes /etc/logindevperm provides a superset of the functionality provided by /etc/fbtab in SunOS 4.x releases.
<table>
<thead>
<tr>
<th><strong>NAME</strong></th>
<th>loginlog – log of failed login attempts</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DESCRIPTION</strong></td>
<td>After five unsuccessful login attempts, all the attempts are logged in the file /var/adm/loginlog. This file contains one record for each failed attempt. Each record contains the login name, tty specification, and time.</td>
</tr>
<tr>
<td></td>
<td>This is an ASCII file. Each field within each entry is separated from the next by a colon. Each entry is separated from the next by a new-line.</td>
</tr>
<tr>
<td></td>
<td>By default, loginlog does not exist, so no logging is done. To enable logging, the log file must be created with read and write permission for owner only. Owner must be root and group must be sys.</td>
</tr>
<tr>
<td><strong>FILES</strong></td>
<td>/var/adm/loginlog</td>
</tr>
<tr>
<td><strong>SEE ALSO</strong></td>
<td>login(1), passwd(1)</td>
</tr>
</tbody>
</table>
The file `/etc/lutab` is a list of the boot environments (BEs) configured on a system. There are two entries for each BE. These entries have the following form:

```plaintext
BE_id:BE_name:completion_flag:0
BE_id:root_slice:root_device:1
```

The fields in the `lutab` entries are described as follows:

- **BE_id**: A unique, internally generated id for a BE.
- **BE_name**: The user-assigned name of a BE.
- **completion_flag**: Indicates whether the BE is complete (C) or incomplete (NC). A complete BE is one that is not involved in any copy or upgrade operation. A BE can be activated or compared only when it is complete.
- **root_slice**: Designation of the root slice.
- **root_device**: Device on which the root slice is mounted.

The `lutab` file must not be edited by hand. Any user modification to this file will result in the incorrect operation of live upgrade.

**SEE ALSO**

- `lu(1M)`, `luactivate(1M)`, `lucreate(1M)`, `lucurr(1M)`, `lufslist(1M)`,
- `lustatus(1M)`, `luupgrade(1M)`, `lutab(4)`, `attributes(5)`, `live_upgrade(5)`

**WARNINGS**

The `lutab` file is not a public interface. The format and contents of `lutab` are subject to change. Use `lufslist(1M)` and `lustatus(1M)` to obtain information about BEs.
The `file` command identifies the type of a file using, among other tests, a test for whether the file begins with a certain **magic number**. The `/etc/magic` file specifies what magic numbers are to be tested for, what message to print if a particular magic number is found, and additional information to extract from the file.

Each line of the file specifies a test to perform. A test compares the data starting at a particular offset in the file with a 1-byte, 2-byte, or 4-byte numeric value or a string. If the test succeeds, a message is printed. The line consists of the following fields (separated by tabs):

```
offset type value message
```

- **offset**: A number specifying the offset, in bytes, into the file of the data which is to be tested.
- **type**: The type of the data to be tested. The possible values are:
  - **byte**: A one-byte value.
  - **short**: A two-byte value.
  - **long**: A four-byte value.
  - **string**: A string of bytes.
  
  The types **byte**, **short**, and **long** may optionally be followed by a mask specifier of the form `&number`. If a mask specifier is given, the value is AND'ed with the `number` before any comparisons are done. The `number` is specified in C form. For instance, 13 is decimal, 013 is octal, and 0x13 is hexadecimal.

- **value**: The value to be compared with the value from the file. If the type is numeric, this value is specified in C form. If it is a string, it is specified as a C string with the usual escapes permitted (for instance, `\n` for NEWLINE).

  **Numeric values** may be preceded by a character indicating the operation to be performed. It may be `'='`, to specify that the value from the file must equal the specified value, `'<'`, to specify that the value from the file must be less than the specified value, `'>'`, to specify that the value from the file must be greater than the specified value, `'&'`, to specify that all the bits in the specified value must be set in the value from the file, `'^'`, to specify that at least one of the bits in the specified value must not be set in the value from the file, or `x` to specify that any value will match. If the character is omitted, it is assumed to be `'='`. 

---

**NAME**
magic – file command’s magic number file

**SYNOPSIS**
/etc/magic

**DESCRIPTION**

Each line of the file specifies a test to perform. A test compares the data starting at a particular offset in the file with a 1-byte, 2-byte, or 4-byte numeric value or a string. If the test succeeds, a message is printed. The line consists of the following fields (separated by tabs):

```
offset type value message
```

- **offset**: A number specifying the offset, in bytes, into the file of the data which is to be tested.
- **type**: The type of the data to be tested. The possible values are:
  - **byte**: A one-byte value.
  - **short**: A two-byte value.
  - **long**: A four-byte value.
  - **string**: A string of bytes.

  The types **byte**, **short**, and **long** may optionally be followed by a mask specifier of the form `&number`. If a mask specifier is given, the value is AND'ed with the `number` before any comparisons are done. The `number` is specified in C form. For instance, 13 is decimal, 013 is octal, and 0x13 is hexadecimal.

- **value**: The value to be compared with the value from the file. If the type is numeric, this value is specified in C form. If it is a string, it is specified as a C string with the usual escapes permitted (for instance, `\n` for NEWLINE).

  **Numeric values** may be preceded by a character indicating the operation to be performed. It may be `'='`, to specify that the value from the file must equal the specified value, `'<'`, to specify that the value from the file must be less than the specified value, `'>'`, to specify that the value from the file must be greater than the specified value, `'&'`, to specify that all the bits in the specified value must be set in the value from the file, `'^'`, to specify that at least one of the bits in the specified value must not be set in the value from the file, or `x` to specify that any value will match. If the character is omitted, it is assumed to be `'='`. 

For string values, the byte string from the file must match the specified byte string. The byte string from the file which is matched is the same length as the specified byte string.

message

The message to be printed if the comparison succeeds. If the string contains a printf(3C) format specification, the value from the file (with any specified masking performed) is printed using the message as the format string.

Some file formats contain additional information which is to be printed along with the file type. A line which begins with the character ‘>’ indicates additional tests and messages to be printed. If the test on the line preceding the first line with a ‘>’ succeeds, the tests specified in all the subsequent lines beginning with ‘>’ are performed, and the messages printed if the tests succeed. The next line which does not begin with a ‘>’ terminates this.

FILES
/etc/magic

SEE ALSO
file(1), file(1B), printf(3C)

BUGS
There should be more than one level of subtests, with the level indicated by the number of ‘>’ at the beginning of the line.
mech, qop – mechanism and QOP files

/etc/gss/mech
/etc/gss/qop

The /etc/gss/mech and /etc/gss/qop files contain tables showing installed security mechanisms and the Quality of Protection (QOP) associated with them, respectively. As security mechanisms are installed on the system, entries are added to these two files. Contents of these files may be accessed either manually (for example, with cat(1) or more(1)) or programmatically (with either rpc_gss_get_mechanisms(3NSL) or rpc_gss_get_mech_info(3NSL)).

The /etc/gss/mech file contains four fields:

- mechanism name: ASCII string representing the mechanism.
- object identifier: RPC OID for this mechanism.
- shared library: Shared library which implements the services provided by this mechanism.
- kernel module: Kernel module which implements the services provided by this mechanism.

The /etc/gss/qop file contains three fields:

- QOP string: Name, in ASCII, of this Quality of Protection.
- QOP value: Numeric value by which RPC identifies this QOP.
- mechanism name: ASCII string representing the mechanism with which this QOP is associated.

EXAMPLE 1 A Typical Entry in /etc/gss/mech

This is a typical entry in a /etc/gss/mech file:
kerberosv5 1.2.840.113554.1.2.2 mech_krb5.so kmech_krb5

EXAMPLE 2 A Typical Entry in /etc/gss/qop

This is a typical entry in a /etc/gss/qop file:
GSS_KRB5_CONF_C_QOP_DES 0 kerberosv5

SEE ALSO
rpc(3NSL), rpc_gss_get_mechanisms(3NSL), rpc_gss_get_mech_info(3NSL), rpcsec_gss(3NSL), attributes ONC+ Developer’s Guide
NAME  
mipagent.conf – configuration file for Mobile IP mobility agent

SYNOPSIS  
/etc/inet/mipagent.conf

DESCRIPTION  
/etc/inet/mipagent.conf is the configuration file used to initialize the Mobile IP mobility agent described in mipagent(1M). Three sample configuration files are located in the /etc/inet directory:

/etc/inet/mipagent.conf-sample
/etc/inet/mipagent.conf.ha-sample
/etc/inet/mipagent.conf.fa-sample

Blank lines are ignored. Lines beginning with the hash character (#) are treated as comments. Sections are denoted by identifiers in brackets. Each section can contain multiple attribute-value pairs. The syntax of an attribute-value pair is an identifier, followed by an equal sign (=), followed by a value.

The following sections and attribute-value pairs must be present in /
/etc/inet/mipagent.conf:

[ General ]
This section contains the Version attribute.

Version
Version is required. For the current release of Mobile IP in Solaris, Version must be 1. Consequently, the default value is 1.

[ Advertisements <interface> ]
This section identifies the interfaces that will serve as Mobile IP mobility agents. One or more of the following attribute-value pairs may be found in this section:

AdvLifeTime
Lifetime (in seconds) advertised in the ICMP router discovery portion of an agent advertisement. See RFC 1256. The default value is 300.

RegLifeTime
Lifetime (in seconds) advertised in the mobility extension of an agent advertisement. The default value is 300.

AdvFrequency
The frequency at which agent advertisements are sent and when different entries are aged. This interval must be less than one-third of AdvLifeTime. The default value is 4.

HomeAgent
Indicates if this agent can act as a home agent. The default value is yes.

ForeignAgent
Indicates if this agent can act as a foreign agent. The default value is yes.

PrefixFlags
Enables the prefix length extension. The default value is yes.
NAIExt
   Enables the Network Access Identifier (NAI) extension. The default value is yes.

Challenge
   Enables the foreign agent challenge extension. The default value is no.

ReverseTunnel
   Indicates if this interface supports reverse tunneling as specified in RFC 2344.
   ReverseTunnel can contain one of the following values:
   no or neither Indicates this interface does not support reverse tunneling.
   FA Indicates only the foreign agent supports reverse tunneling.
   HA Indicates only the home agent supports reverse tunneling.
   yes or both Indicates that both foreign and home agents support reverse tunneling as specified in RFC 2344.

   The default value for ReverseTunnel is no.

ReverseTunnelRequired
   Indicates if this interface will require reverse tunneling as specified in RFC 2344.
   ReverseTunnelRequired can contain one of the following values:
   no or neither Indicates this interface will not require reverse tunneling.
   FA Indicates only the foreign agent will require a reverse tunnel.
   HA Indicates only the home agent will require a reverse tunnel.
   yes or both Indicates that both foreign and home agents will require a reverse tunnel.

   The default value for ReverseTunnelRequired is no.

[ GlobalSecurityParameters ]
   This section defines the global security parameters that will be used to authenticate mobile nodes. MN-HA authentication is always enabled. This section may contain one or more of the following attribute-value pairs:

   HA-FAAuth Enables home agent - foreign agent authentication. The default value is yes.
   MN-FAAuth Enables mobile node - foreign agent authentication. The default value is no.
   MaxClockSkew The maximum allowable difference in clocks, in seconds, that will be tolerated. This is used for replay protection. The default value is 300.
   KeyDistribution This attribute defines where keys are found. The default for this Version of Solaris Mobile IP software is files.
These sections define multiple Security Parameter Indices (SPIs). One section is required for each security context. These SPI values are used in the Address section to define the security used for a particular mobile node or agent. In this section, both the Key and ReplayMethod attributes must be present.

**Key**
The hexadecimal representation of the key used for authentication.

**ReplayMethod**
The replay method. Possible values are timestamps or none.

These sections define address pools for dynamically assigned IP addresses. The Start and Length attributes both must be present.

**Start**
The beginning range of the IP address from which to allocate an IP address in dotted quad notation.

**Length**
The length of the IP address range.

This section defines the security policy used for each host for which an NAI or IP address is specified in the section header. The keyword node-default is used to create a single entry that can be used by any mobile node that has the correct SPI and associated keying information. This section specifies the SPI, and in the case of mobile nodes, pool numbers for NAI addresses.

**Type**
Indicates whether the address entry specifies a mobile node or a mobility agent.

**SPI**
The SPI used for this Address.

**Pool**
The Pool used for this NAI address. The Pool keyword may only be present if the Type operand is set to mobile node.

### EXAMPLES

**EXAMPLE 1** Configuration for Providing Mobility Services on One Interface

The following example shows the configuration file for a mobility agent that provides mobility services on one interface (le0). The mobility agent acts both as a home agent as well as a foreign agent on that interface. It includes the prefix length in its advertisements. Its home and foreign agent functions support reverse tunneling, but only the foreign agent requires that a reverse tunnel be configured. The mobility agent provides home agent services to three mobile nodes: 192.168.10.17, 192.168.10.18, and the NAI address user@defaultdomain.com.
EXAMPLE 1 Configuration for Providing Mobility Services on One Interface  (Continued)

With the first mobile node, the agent uses an SPI of 257 (decimal) and a shared secret key that is six bytes long containing alternate bytes that are 0 and 255 (decimal). For the second mobile node, the SPI is 541 (decimal), the key is 10 bytes, and it contains the decimal values 11 through 20 in those bytes. The first mobile node uses no replay protection, and the second uses timestamps. The third mobile node uses NAI and gets its address from Pool 1.

The mobile node will also need to be configured with the same security association that is specified in the home agent’s configuration file.

```bash
# start of file
[ General ]
Version = 1

[ Advertisements le0 ]
AdvLifeTime = 200
RegLifetime = 200
AdvFrequency = 5
AdvertiseOnBcast = yes
HomeAgent = yes
ForeignAgent = yes
PrefixFlags = yes
ReverseTunnel = both
ReverseTunnelRequired = FA

[ GlobalSecurityParameters ]
HA-FAAuth = no
MN-FAAuth = no
KeyDistribution = files

[ SPI 257 ]
Key = 00ff00ff00ff
ReplayMethod = none

[ SPI 541 ]
Key = 0b0c0d0e0f1011121314
ReplayMethod = timestamps

[ Pool 1 ]
Start = 192.168.167.1
Length = 250

[ Address 192.168.10.17 ]
Type = node
SPI = 257

[ Address 192.168.10.18 ]
Type = node
SPI = 541

[ Address user@defaultdomain.com ]
Type = node
```

File Formats  215
EXAMPLE 1 Configuration for Providing Mobility Services on One Interface  (Continued)

SPI = 541
Pool = 1

[ Address node-default ]
Type = node
SPI = 541
Pool = 1
#end of file

FILES
/etc/inet/mipagent.conf
  Configuration file for Mobile IP mobility agent
/etc/inet/mipagent.conf-sample
  Sample configuration file for mobility agents.
/etc/inet/mipagent.conf.ha-sample
  Sample configuration file for home agent functionality.
/etc/inet/mipagent.conf.fa-sample
  Sample configuration file for foreign agent functionality.

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

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

SEE ALSO
mipagent(1M), mipagentconfig(1M), attributes(5)


NOTES
The base Mobile IP protocol (RFC 2002) does not address the problem of scalable key distribution and treats key distribution as an orthogonal issue. The Solaris Mobile IP software utilizes manually configured keys only, specified in a configuration file.
### NAME
mnttab – mounted file system table

### DESCRIPTION
The file `/etc/mnttab` is really a file system that provides read-only access to the table of mounted file systems for the current host. `/etc/mnttab` is read by programs using the routines described in `getmntent(3C)`. Mounting a file system adds an entry to this table. Unmounting removes an entry from this table. Remounting a file system causes the information in the mounted file system table to be updated to reflect any changes caused by the remount. The list is maintained by the kernel in order of mount time. That is, the first mounted file system is first in the list and the most recently mounted file system is last. When mounted on a mount point the file system appears as a regular file containing the current `mnttab` information.

Each entry is a line of fields separated by spaces in the form:

```
special  mount_point  fstype  options  time
```

where

- **special**: The name of the resource to be mounted.
- **mount_point**: The pathname of the directory on which the filesystem is mounted.
- **fstype**: The file system type of the mounted file system.
- **options**: The mount options. (See respective mount file system man page in **SEE ALSO**.)
- **time**: The time at which the file system was mounted.

Examples of entries for the `special` field include the pathname of a block-special device, the name of a remote file system in the form of `host:pathname`, or the name of a swap file (for example, a file made with `mkfile(1M)`).

### IOCTLS
The following `ioctl(2)` calls are supported:

- **MNTIOC_NMOUNTS**: Returns the count of mounted resources in the current snapshot in the `uint32_t` pointed to by `arg`.
- **MNTIOC_GETDEVLIST**: Returns an array of `uint32_t`'s that is twice as long as the length returned by `MNTIOC_NMOUNTS`. Each pair of numbers is the major and minor device number for the file system at the corresponding line in the current `/etc/mnttab` snapshot. `arg` points to the memory buffer to receive the device number information.
- **MNTIOC_SETTAG**: Sets a tag word into the options list for a mounted file system. A tag is a notation that will appear in the options string of a mounted file system but it is not recognized or interpreted by the file system code. `arg` points to a filled in `mnttagdesc` structure, as shown in the following example:
uint_t mtd_major; /* major number for mounted fs */
uint_t mtd_minor; /* minor number for mounted fs */
char *mtd_mntpt; /* mount point of file system */
char *mtd_tag; /* tag to set/clear */

If the tag already exists then it is marked as set but not re-added. Tags can be at most MAX_MNTOPT_TAG long.

MNTIOC_CLRTAG Marks a tag in the options list for a mounted file system as not set. arg points to the same structure as MNTIOC_SETTAG, which identifies the file system and tag to be cleared.

ERRORS

EFAULT The arg pointer in an MNTIOC_ioctl call pointed to an inaccessible memory location or a character pointer in a mnttagdesc structure pointed to an inaccessible memory location.

EINVAL The tag specified in a MNTIOC_SETTAG call already exists as a file system option, or the tag specified in a MNTIOC_CLRTAG call does not exist.

ENAMETOOLONG The tag specified in a MNTIOC_SETTAG call is too long or the tag would make the total length of the option string for the mounted file system too long.

WARNINGS

The mnttab file system provides the previously undocumented dev=xxx option in the option string for each mounted file system. This is provided for legacy applications that might have been using the dev=information option.

Using dev=option in applications is strongly discouraged. The device number string represents a 32-bit quantity and might not contain correct information in 64-bit environments.

Applications requiring device number information for mounted file systems should use the getextmntent(3C) interface, which functions properly in either 32- or 64-bit environments.

FILES

/etc/mnttab
  Usual mount point for mnttab file system
/usr/include/sys/mntio.h
  Header file that contains IOCTL definitions

SEE ALSO

mkfile(1M), mount_cachefs(1M), mount_hafs(1M), mount_nfs(1M),
mount_pcfs(1M), mount_ufs(1M), mount(1M), ioctl(2), read(2), poll(2),
stat(2), getextmntent(3C)

NOTES

The snapshot of the mnttab information is taken any time a read(2) is performed at offset 0 (the beginning) of the mnttab file. The file modification time returned by stat(2) for the mnttab file is the time of the last change to mounted file system
information. A poll(2) system call requesting a POLLRDBAND event can be used to block and wait for the system’s mounted file system information to be different from the most recent snapshot since the mnttab file was opened.
BIND version 8 is a much more configurable version than previous releases of BIND. New areas of configuration include access control lists and categorized logging. Many options that previously applied to all zones can now be used selectively. The new configuration file format in named.conf incorporates these features and allows for consideration of future configuration needs.

A BIND 8 configuration file consists of two general features, statements and comments.

All statements end with a semicolon. Many statements allow substatements, which also terminate with a semicolon. BIND 8 supports the following statements:

- **logging**: Specifies what the server logs, and where the log messages are sent.
- **options**: Controls global server configuration options and sets defaults for other statements.
- **zone**: Defines a zone.
- **acl**: Defines a named IP address matching list, for access control and other uses.
- **key**: Specifies key information for use in authentication and authorization.
- **trusted-keys**: Defines DNSSEC keys that are preconfigured into the server and implicitly trusted.
- **server**: Sets certain configuration options for individual remote servers.
- **controls**: Declares control channels to be used by the ndc(1M) utility.
- **include**: Includes another file.

The logging and options statements may only occur once per configuration, while the rest may appear numerous times. Further detail on each statement is provided in individual sections below.

Comments may appear anywhere that whitespace may appear in a BIND configuration file. To appeal to programmers of all kinds, they can be written in C, C++, shell or perl constructs.

C-style comments start with the two characters / * (slash, star) and end with */ (star, slash). Because comments are completely delimited by these characters, they can be used to comment either a portion of a line or to span multiple lines.

C-style comments cannot be nested. For example, the following is not valid because the entire comment ends with the first */: 

```
```
/* This is the start of a comment.
   This is still part of the comment.
*/
/* This is an incorrect attempt at nesting a comment. */
 This is no longer in any comment. */

C++ style comments start with the two characters // (slash, slash) and continue to the end of the physical line. They cannot be continued across multiple physical lines. To have one logical comment span multiple lines, each line must use the // pair. For example:

// This is the start of a comment. The next line
// is a new comment, even though it is logically
// part of the previous comment. Shell-style or perl-style comments start with the character # (hash or pound or number or octothorpe or whatever) and like C++ comments, continue to the end of the physical line. For example:

# This is the start of a comment. The next line
# is a new comment, even though it is logically
# part of the previous comment. BIND 4.9.x configuration files can be converted to the new format by using named-bootconf(1M).

The elements described below are used throughout the BIND configuration file documentation. Elements which are only associated with one statement are described only in the section describing that statement.

- **acl_name**
  The name of an address_match_list as defined by the acl statement.

- **address_match_list**
  A list of one or more ip_addr, ip_prefix, key_id, or acl_name elements, as described in the ADDRESS MATCH LISTS section.

- **dotted-decimal**
  One or more integers valued 0 through 255 separated only by dots ("."), such as 123, 45.67 or 89.123.45.67.

- **domain_name**
  A quoted string which will be used as a DNS name, for example "my.test.domain".

- **path_name**
  A quoted string which will be used as a pathname, such as "zones/master/my.test.domain".

- **ip_add**
  An IP address in with exactly four elements in dotted-decimal notation.

- **ip_port**
  An IP port number. number is limited to 0 through 65535, with values below 1024 typically restricted to root-owned processes. In some cases an asterisk ("*") character can be used as a placeholder to select a random high-numbered port.

- **ip_prefix**
  IP network specified in dotted-decimal form, followed by "/" and then the number of bits in the netmask. For
example, 127/8 is the network 127.0.0.0 with netmask 255.0.0.0. 1.2.3.0/28 is network 1.2.3.0 with netmask 255.255.255.240.

**key_name**
A string representing the name of a shared key, to be used for transaction security.

**number**
A non-negative integer with an entire range limited by the range of a C language signed integer (2,147,483,647 on a machine with 32 bit integers). Its acceptable value might be further limited by the context in which it is used.

**size_spec**
number, the word unlimited, or the word default.

The maximum value of size_spec is that of unsigned long integers on the machine. unlimited requests unlimited use, or the maximum available amount. default uses the limit that was in force when the server was started.

A number can optionally be followed by a scaling factor: K or k for kilobytes, M or m for megabytes, and G or g for gigabytes, which scale by 1024, 1024*1024, and 1024*1024*1024 respectively.

Integer storage overflow is currently silently ignored during conversion of scaled values, resulting in values less than intended, possibly even negative. Using unlimited is the best way to safely set a really large number.

**yes_or_no**
Either yes or no. The words true and false are also accepted, as are the numbers 1 and 0.

**ADDRESS MATCH LISTS**

**Syntax**

```
address_match_list = 1*address_match_element

address_match_element = [ "!" ] ( address_match_list / ip_address / ip_prefix / acl_name / "key " key_id ) ";
```

**Definition and Usage**

Address match lists are primarily used to determine access control for various server operations. They are also used to define priorities for querying other name servers and to set the addresses on which in.named(1M) in.named will listen for queries. The elements which constitute an address match list can be any of the following:

- an *ip-address* (in dotted-decimal notation)
- an *ip-prefix* (in the ’/’-notation)
- A *key_id*, as defined by the key statement
- the name of an address match list previously defined with the acl statement
- or, another `address_match_list`.

Elements can be negated with a leading exclamation mark (`''!''`), and the match list names any, `none`, `localhost` and `localnets` are predefined. More information on those names can be found in the description of the acl statement.

The addition of the key clause made the name of this syntactic element something of a misnomer, since security keys can be used to validate access without regard to a host or network address. Nonetheless, the term “address match list” is still used throughout the documentation.

When a given IP address or prefix is compared to an address match list the list is traversed, in order, until an element matches. The interpretation of a match depends on whether the list is being used for access control, for defining listen-on ports, or as a topology, and whether the element is negated.

When used as an access control list, a non-negated match allows access, and a negated match denies access. If there is no match at all in the list, access is denied. The clauses `allow-query`, `allow-transfer`, `allow-update`, `allow-recursion` and `blackhole` all use address match lists like this. Similarly, the `listen-on` option will cause the server to not accept queries on any of the machine's addresses that do not match the list.

When used with the topology option, a non-negated match returns a distance based on its position on the list. The closer the match is to the start of the list, the shorter the distance is between it and the server. A negated match will be assigned the maximum distance from the server. If there is no match, the address will get a distance which is further than any non-negated list element, and closer than any negated element.

Because of the first-match aspect of the algorithm, an element that defines a subset of another element in the list should come before the broader element, regardless of whether either is negated. For example, in

```
1.2.3/24; !1.2.3.13
```

the 1.2.3.13 element is completely useless, because the algorithm will match any lookup for 1.2.3.13 to the 1.2.3/24 element. Using

```
!1.2.3.13; 1.2.3/24
```

fixes that problem by having 1.2.3.13 blocked by the negation but all other 1.2.3.* hosts fall through.
The logging statement configures a wide variety of logging options for the name server. Its channel phrase associates output methods, format options and severity levels with a name that can then be used with the category phrase to select how various classes of messages are logged.

Only one logging statement is used to define as many channels and categories as are wanted. If there are multiple logging statements in a configuration, the first defined determines the logging, and warnings are issued for the others. If there is no logging statement, the logging configuration will be:

```
logging {
    category default { default_syslog; default_debug; }; 
    category panic { default_syslog; default_stderr; }
    category packet { default_debug; }
    category eventlib { default_debug; }
};
```

The logging configuration is established as soon as the logging statement is parsed. If you want to redirect messages about processing of the entire configuration file, the logging statement must appear first. Even if you do not redirect configuration file parsing messages, we recommend always putting the logging statement first so that this rule need not be consciously recalled if you ever do want to relocate the parser’s messages.

All log output goes to one or more "channels." You can make as many of them as you want.

Every channel definition must include a clause that says whether messages selected for the channel go to a file, to a particular syslog(3C) facility, or are discarded. It can optionally also limit the message severity level that will be accepted by the channel (the default is info), and whether to include a time stamp generated by in.named(1M), the category name, or severity level. The default is not to include any of those three.

The word null as the destination option for the channel will cause all messages sent to it to be discarded. Other options for the channel are meaningless.
The file clause can include limitations both on how large the file is allowed to become and how many versions of the file will be saved each time the file is opened.

The size option for files is simply a hard ceiling on log growth. If the file ever exceeds the size, then in.named will not write anything more to it until the file is reopened. That the size is exceeded does not automatically trigger a reopen. The default behavior does not limit the size of the file.

If you use the version logfile option, then in.named will retain the backup versions of the file by renaming them when it opens them. For example, if you choose to keep 3 old versions of the file lamers.log then just before it is opened lamers.log.1 is renamed to lamers.log.2, lamers.log.0 is renamed to lamers.log.1, and lamers.log is renamed to lamers.log.0. No rolled versions are kept by default. Any existing log file is simply appended. The unlimited keyword is synonymous with 99 in current BIND releases. Example usage of size and versions options:

```
channel an_example_level {
    file "lamers.log" versions 3 size 20m;
    print-time yes;
    print-category yes;
}
```

The argument for the syslog() clause is a syslog() facility as described in the syslog(3C) manual page. How syslogd(1M) will handle messages sent to this facility is described in the syslog.conf(4).

The severity clause works like the priority levels for syslog(), except that they can also be used if you are writing straight to a file rather than using syslog(). Messages which are not at least of the severity level given will not be selected for the channel; messages of higher severity levels will be accepted.

If you are using syslog(), then the syslog.conf priorities will also determine what eventually passes through. For example, defining a channel facility and severity as daemon and debug but only logging daemon warnings by means of syslog.conf will cause messages of severity info and notice to be dropped. If the situation were reversed, with in.named writing messages of only warning or higher, then syslogd will print all messages it receives from the channel.

The server can supply extensive debugging information when it is in debugging mode. If the server’s global debug level is greater than zero, then the debugging mode will be active. The global debug level is set either by starting the in.named server with the -d flag followed by a positive integer, or by sending the running server the SIGUSR1 signal (for example, by using ndc trace). The global debug level can be set to zero and debugging mode turned off, by sending the server the SIGUSR2 signal (as with ndc notrace). All debugging messages in the server have a debug level, and higher debug levels give more more detailed output. Channels that specify a specific debug severity, for example:

```
channel specific_debug_level {
    file "foo";
    severity debug 3;
}
```
will get debugging output of level 3 or less any time the server is in debugging mode, regardless of the global debugging level. Channels with dynamic severity use the server's global level to determine what messages to print.

If print-time has been turned on, then the date and time will be logged. print-time may be specified for a syslog() channel, but is usually unnecessary since syslog() also prints the date and time. If print-category is requested, then the category of the message will be logged as well. Finally, if print-severity is on, then the severity level of the message will be logged. The print- options may be used in any combination, and will always be printed in the following order: time, category, severity. Here is an example where all three print- options are on:

print-time has been turned on, then the date and time will be logged. print-time may be specified for a syslog() channel, but is usually unnecessary since syslog() also prints the date and time. If print-category is requested, then the category of the message will be logged as well. Finally, if print-severity is on, then the severity level of the message will be logged. The print- options may be used in any combination, and will always be printed in the following order: time, category, severity. Here is an example where all three print- options are on:


There are four predefined channels that are used for default logging in.named(1M). How they are used is described in the next section, The Category Phrase.

channel default_syslog {
    syslog daemon; # send to syslog's daemon facility
    severity info; # only send priority info and higher
};

channel default_debug {
    file "named.run"; # write to named.run in the working directory
    severity dynamic; # log at the server's current debug level
};

channel default_stderr {
    file "stderr"; # this is illustrative only; there's currently
    # no way of specifying an internal file
    severity info; # descriptor in the configuration language.
};

channel null {
    null; # toss anything sent to this channel
};

Once a channel is defined, it cannot be redefined. Thus you cannot alter the built-in channels directly, but you can modify the default logging by pointing categories at channels you have defined.

There are many categories, so you can send the logs you want to see wherever you want, without seeing logs you do not want. If you do not specify a list of channels for a category, then log messages in that category will be sent to the default category instead. If you do not specify a default category, the following "default default" is used:

category default { default_syslog; default_debug; }

To log security events to a file but also keep the default logging behavior, specify the following:
channel my_security_channel {
    file "my_security_file";
    severity info;
};
category security { my_security_channel;
    default_syslog; default_debug; }; To discard all messages in a category, specify the null channel:
category lame-servers { null; }
category cname { null; }

The following categories are available:

default The catch-all. Many things still are not classified into categories, and they all end up here. Also, if you don’t specify any channels for a category, the default category is used instead. If you do not define the default category, the following definition is used:
category default { default_syslog; default_debug; };

config High-level configuration file processing.
parser Low-level configuration file processing.
queries A short log message is generated for every query the server receives.
lame-servers Messages like “Lame server on…”
statistics Statistics.
panic If the server has to shut itself down due to an internal problem, it will log the problem in this category as well as in the problem’s native category. If you do not define the panic category, the following definition is used:
category panic { default_syslog; default_stderr; };

update Dynamic updates.
ncache Negative caching.
xfer-in Zone transfers the server is receiving.
xfer-out Zone transfers the server is sending
db All database operations.
eventlib Debugging information from the event system. Only one channel may be specified for this category, and it must be a file channel. If you do not define the eventlib category, the following definition is used:
category eventlib { default_debug; };

category packet { default_debug; };

Dumps of packets received and sent. Only one channel may be specified for this category, and it must be a file channel. If you do not define the packet category, the following definition is used:

notify

The Notify protocol.

cname

Messages like “... points to a CNAME”.

security

Approved or unapproved requests.

os

Operating system problems.

insist

Internal consistency check failures.

maintenance

Periodic maintenance events.

load

Load.

response-checks

Messages arising from response checking, such as “Malformed response ...”, “wrong ans. name ...”, “unrelated additional info ...”, “invalid RR type ...”, and “bad referral ...”.

options {
    [ version version_string; ]
    [ directory path_name; ]
    [ named-xfer path_name; ]
    [ dump-file path_name; ]
    [ memstatistics-file path_name; ]
    [ pid-file path_name; ]
    [ statistics-file path_name; ]
    [ auth-nxdomain yes_or_no; ]
    [ deallocate-on-exit yes_or_no; ]
    [ dialup yes_or_no; ]
    [ fake-iquery yes_or_no; ]
    [ fetch-glue yes_or_no; ]
    [ has-old-clients yes_or_no; ]
    [ host-statistics yes_or_no; ]
    [ multiple-cnames yes_or_no; ]
    [ notify yes_or_no; ]
    [ recursion yes_or_no; ]
    [ rfc2308-type1 yes_or_no; ]
    [ use-id-pool yes_or_no; ]
    [ treat-cr-as-space yes_or_no; ]
    [ also-notify yes_or_no; ]
    [ forward ( only | first ); ]
    [ forwarders { [ in_addr ; [ in_addr ; ... ] ] }; ]
    [ check-names ( master | slave | response ) ( warn | fail | ignore); ]
The `options` statement sets up global options to be used by BIND. This statement may appear only once in a configuration file. If more than one occurrence is found, the first occurrence determines the options used, and a warning will be generated. If there is no options statement, an options block with each option set to its default will be used.

### Definition and Usage

**Pathnames**

- **version**
  - The version the server should report by means of the `ndc(1M)` command or by means of a query of name `version.bind` in class chaos. The default is the real version number of the server.

- **directory**
  - The working directory of the server. Any non-absolute pathnames in the configuration file will be taken as relative to this directory. The default location for most server output files, for example, `named.run`, is this directory. If a directory is not specified, the working directory defaults to `""`, the directory from which the server was started. The directory specified should be an absolute path.
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>named-xfer</strong></td>
<td>The pathname to the <code>named-xfer</code> program that the server uses for inbound zone transfers. If not specified, the default is <code>/usr/sbin/named-xfer</code>.</td>
</tr>
<tr>
<td><strong>dump-file</strong></td>
<td>The pathname of the file to which the server dumps the database when it receives a SIGINT signal, for example, as sent by <code>ndc dump</code>. If not specified, the default is <code>named_dump.db</code>.</td>
</tr>
<tr>
<td><strong>memstatistics-file</strong></td>
<td>The pathname of the file the server writes memory usage statistics to on exit, if <code>deallocate-on-exit</code> is yes. If not specified, the default is <code>named.memstats</code>.</td>
</tr>
<tr>
<td><strong>pid-file</strong></td>
<td>The pathname of the file in which the server writes its process ID. If not specified, the default is <code>/var/run/named.pid</code>.</td>
</tr>
<tr>
<td><strong>statistics-file</strong></td>
<td>The pathname of the file the server appends statistics to when it receives a SIGILL signal. If not specified, the default is <code>named.stats</code>.</td>
</tr>
<tr>
<td><strong>auth-nxdomain</strong></td>
<td>If the value is yes, then the AA bit is always set on NXDOMAIN responses, even if the server is not actually authoritative. The default is yes. Do not turn off <code>auth-nxdomain</code> unless you are sure you know what you are doing, as some older software will not like it.</td>
</tr>
<tr>
<td><strong>deallocate-on-exit</strong></td>
<td>If the value is yes, then when the server exits it will painstakingly deallocate every object it allocated, and then write a memory usage report to the <code>memstatistics-file</code>. The default is no because it is faster to let the operating system clean up. <code>deallocate-on-exit</code> is handy for detecting memory leaks.</td>
</tr>
<tr>
<td><strong>dialup</strong></td>
<td>If the value is yes, then the server treats all zones as if they are doing zone transfers across a dial on a demand dialup link, which can be brought up by traffic originating from this server. This has different effects according to the zone type. It concentrates the zone maintenance so that it all happens in a short interval, once every <code>heartbeat-interval</code> and hopefully, during the one call. It also suppresses some of the normal zone maintenance traffic. The default is no. The <code>dialup</code> option may also be specified in the zone statement, in which case it overrides the options <code>dialup</code> statement.</td>
</tr>
</tbody>
</table>
If the zone is a master then the server will send out NOTIFY request to all the slaves. This will trigger the zone up to date checking in the slave, providing the slave supports NOTIFY, and allowing the slave to verify the zone while they call us up. If the zone is a slave or stub, then the server will suppress the regular zone up to date queries, and only perform them when the heartbeat-interval expires.

fake-iquery
If yes, the server will simulate the obsolete DNS query type IQUERY. The default is no.

fetch-glue
If yes (the default), the server will fetch "glue" resource records it does not have when it constructs the additional data section of a response. fetch-glue no can be used in conjunction with recursion no to prevent the server's cache from growing or becoming corrupted. However, it requires more work from the client.

has-old-clients
Setting the option to yes is equivalent to setting the following three options: auth-nxdomain yes, maintain-ixfr-base yes, and rfc2308-type1 no. has-old-clients with auth-nxdomain, maintain-ixfr-base, and rfc2308-type1 is order dependant.

host-statistics
If yes, then statistics are kept for every host with which the name server interacts. The default is no. Turning on host-statistics can consume huge amounts of memory.

multiple-cnames
If yes, then multiple CNAME resource records will be allowed for a domain name. The default is no. Allowing multiple CNAME records is against standards and is not recommended. Multiple CNAME support is available because previous versions of BIND allowed multiple CNAME records, and these records have been used for load balancing by a number of sites.

notify
If yes (the default), DNS NOTIFY messages are sent when a zone for which the server is authoritative changes. The use of NOTIFY speeds convergence between the master and its slaves. Slave servers that receive a NOTIFY message and understand it will contact the master server for the zone and see if they need to do a zone transfer. If they do, they will initiate it immediately. The notify option may also be specified in the zone statement, in which case it overrides the options notify statement.
### recursion
If yes, and a DNS query requests recursion, then the server will attempt to do all the work required to answer the query. If recursion is not on, the server will return a referral to the client if it does not know the answer. The default is yes. See also fetch-glue above.

### rfc2308-type1
If yes, the server will send NS records along with the SOA record for negative answers. If you have an old BIND server using you as a forwarder, which does not understand negative answers that contain both SOA and NS records, or you have an old version of sendmail(1M), set this to no. The correct fix is to upgrade the broken server or sendmail. The default is no.

### use-id-pool
If yes, the server will keep track of its own outstanding query ID’s to avoid duplication and increase randomness. As a result, the server will consume 128KB more memory. The default is no.

### treat-cr-as-space
If yes, the server will treat CR characters the same way it treats a space or tab. This may be necessary when loading zone files on a UNIX system that were generated on either an NT or a DOS machine. The default is no.

### Also-Notify

<table>
<thead>
<tr>
<th>also-notify</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defines a global list of IP addresses that also get sent NOTIFY messages whenever a fresh copy of the zone is loaded. This helps to ensure that copies of the zones will quickly converge on “stealth” servers. If an also-notify list is given in a zone statement, it will override the options also-notify statement. When a zone notify statement is set to no, the IP addresses in the global also-notify list will not get sent NOTIFY messages for that zone. The default is the empty list (no global notification list).</td>
</tr>
</tbody>
</table>

### Forwarding
The forwarding facility can be used to create a large site-wide cache on a few servers. This reduces traffic over links to external name servers. It can also be used to allow queries by servers that do not have direct access to the Internet but wish to look up exterior names anyway. Forwarding occurs only on those queries for which the server is not authoritative and does not have the answer in its cache.

<table>
<thead>
<tr>
<th>forward</th>
</tr>
</thead>
<tbody>
<tr>
<td>This option is only meaningful if the forwarders list is not empty. A value of first, the default, causes the server to query the forwarders first. If the forwarders do not answer the question, the server will then look for the answer itself. If only is specified, the server will only query the forwarders.</td>
</tr>
</tbody>
</table>
forwarders

Specifies the IP addresses to be used for forwarding.
The default is the empty list (no forwarding).

Forwarding can also be configured on a per-zone basis, allowing for the global forwarding options to be overridden in a variety of ways. You can set particular zones to use different forwarders, have different forward only or forward first behavior, or not forward at all. See THE ZONE STATEMENT section for more information.

Future versions of BIND 8 may provide a more powerful forwarding system. The syntax described above will continue to be supported.

Name Checking

The server can check domain names based upon their expected client contexts. For example, a domain name used as a hostname can be checked for compliance with the RFCs that define valid hostnames.

Three checking methods are available:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ignore</td>
<td>No checking is done.</td>
</tr>
<tr>
<td>warn</td>
<td>Names are checked against their expected client contexts. Invalid names are logged, but processing continues normally.</td>
</tr>
<tr>
<td>fail</td>
<td>Names are checked against their expected client contexts. Invalid names are logged, and the offending data is rejected.</td>
</tr>
</tbody>
</table>

The server can check names three areas: master zone files, slave zone files, and responses to queries the server has initiated. If check-names response fail has been specified, and to answer the client’s question would require sending an invalid name to the client, the server will send a REFUSED response code to the client.

The defaults are:

- check-names master fail
- check-names slave warn
- check-names response ignore

check-names may also be specified in the zone statement, in which case it overrides the options check-names statement. When used in a zone statement, the area is not specified, as it can be deduced from the zone type.

Access Control

Access to the server can be restricted based on the IP address of the requesting system or by means of shared secret keys. See ADDRESS MATCH LISTS for details on how to specify access criteria.

allow-query

Specifies which hosts are allowed to ask ordinary questions.

allow-query may also be specified in the zone statement, in which case it overrides the options allow-query statement. If not specified, the default is:
allow-recursion
Specifies which hosts are allowed to ask recursive questions.
allow-recursion may also be specified in the zone statement, in which case it overrides the options allow-recursion statement. If not specified, the default is to allow recursive queries from all hosts.

allow-transfer
Specifies which hosts are allowed to receive zone transfers from the server. allow-transfer may also be specified in the zone statement, in which case it overrides the options allow-transfer statement. If not specified, the default is to allow transfers from all hosts.

blackhole
Specifies a list of addresses that the server will not accept queries from or use to resolve a query. Queries from these addresses will not receive a response.

The interfaces and ports that the server will answer queries from may be specified using the listen-on option. listen-on takes an optional port and an address match list. The server will listen on all interfaces allowed by the address match list. If a port is not specified, port 53 will be used.

Multiple listen-on statements are allowed. For example,

```
listen-on { 5.6.7.8; };  
listen-on port 1234 { !1.2.3.4; 1.2/16; }; 
```

will enable the name server on port 53 for the IP address 5.6.7.8, and on port 1234 of an address on the machine in net 1.2 that is not 1.2.3.4.

If no listen-on is specified, the server will listen on port 53 on all interfaces.

If the server does not know the answer to a question, it will query other name servers. query-source specifies the address and port used for such queries. If address is * or is omitted, a wildcard IP address (INADDR_ANY) will be used. If port is * or is omitted, a random unprivileged port will be used. The default is

```
query-source address * port *; 
```

query-source currently applies only to UDP queries; TCP queries always use a wildcard IP address and a random unprivileged port.
<table>
<thead>
<tr>
<th>Zone Transfers</th>
<th>max-transfer-time-in</th>
<th>Inbound zone transfers (named-xfer processes) running longer than max-transfer-time-in minutes will be terminated. The default value for max-transfer-time-in is 120 minutes (2 hours).</th>
</tr>
</thead>
<tbody>
<tr>
<td>transfer-format</td>
<td></td>
<td>The server supports two zone transfer methods. one-answer uses one DNS message per resource record transferred. many-answers packs as many resource records as possible into a message. many-answers is more efficient, but is only known to be understood by BIND 8.1 and patched versions of BIND 4.9.5. The default is one-answer. transfer-format may be overridden on a per-server basis by using the server statement.</td>
</tr>
<tr>
<td>transfers-in</td>
<td></td>
<td>The maximum number of inbound zone transfers that can be running concurrently. The default value is 10. Increasing transfers-in may speed up the convergence of slave zones, but it also may increase the load on the local system.</td>
</tr>
<tr>
<td>transfers-out</td>
<td></td>
<td>This option will be used in the future to limit the number of concurrent outbound zone transfers. It is checked for syntax, but is otherwise ignored.</td>
</tr>
<tr>
<td>transfers-per-ns</td>
<td></td>
<td>The maximum number of inbound zone transfers (named-xfer processes) that can be concurrently transferred from a given remote name server. The default value is 2. Increasing transfers-per-ns may speed up the convergence of slave zones, but it also may increase the load on the remote name server. transfers-per-ns may be overridden on a per-server basis by using the transfers phrase of the server statement.</td>
</tr>
<tr>
<td>transfer-source</td>
<td></td>
<td>transfer-source determines which local address will be bound to the TCP connection used to fetch all zones transferred inbound by the server. If not set, it defaults to a system controlled value which will usually be the address of the interface “closest to” the remote end. This address must appear in the remote end’s allow-transfer option for the zones being transferred, if one is specified. This statement sets the transfer-source for all zones, but can be overridden on a per-zone basis by including a transfer-source statement within the zone block in the configuration file.</td>
</tr>
</tbody>
</table>
The server’s usage of many system resources can be limited. Some operating systems do not support some of the limits. On such systems, a warning will be issued if the unsupported limit is used. Some operating systems do not support resource limits, and on these systems a

```
set resource limits on this system will be logged.
```

Scaled values are allowed when specifying resource limits. For example, 1G can be used instead of 1073741824 to specify a limit of one gigabyte. Other values include: unlimited requests, unlimited use, or the maximum available amount. The value default uses the limit that was in force when the server was started. See the definition of size_spec for more details.

coresize The maximum size of a core dump. The default value is default.
datasize The maximum amount of data memory the server may use. The default value is default.
files The maximum number of files the server may have open concurrently. The default value is unlimited. Note that on some operating systems the server cannot set an unlimited value and cannot determine the maximum number of open files the kernel can support. On such systems, choosing unlimited will cause the server to use the larger of the rlim_max from getrlimitRLIMIT_NOFILE() and the value returned by sysconf_SC_OPEN_MAX(). If the actual kernel limit is larger than this value, use limit files to specify the limit explicitly.

```
max-ixfr-log-size The max-ixfr-log-size will be used in a future release of the server to limit the size of the transaction log kept for Incremental Zone Transfer.
```

stacksize The maximum amount of stack memory the server may use. The default value is default.

```
Periodic Task Intervals
```

cleaning-interval The server will remove expired resource records from the cache every cleaning-interval minutes. The default is 60 minutes. If set to 0, no periodic cleaning will occur.

heartbeat-interval The server will perform zone maintenance tasks for all zones marked dialup yes whenever this interval expires. The default is 60 minutes. Reasonable values are up to 1 day (1440 minutes). If set to 0, no zone maintenance for these zones will occur.

```
interface-interval The server will scan the network interface list every interface-interval minutes. The default is 60
```

minutes. If set to 0, interface scanning will only occur when the configuration file is loaded. After the scan, listeners will be started on any new interfaces, provided they are allowed by the listen-on configuration. Listeners on interfaces that have gone away will be cleaned up.

statistics-interval
Name server statistics will be logged every statistics-interval minutes. The default is 60. If set to 0, no statistics will be logged.

Topology
All other things being equal, when the server chooses a name server to query from a list of name servers, it prefers the one that is topologically closest to itself. The topology statement takes an address match list and interprets it in a special way. Each top-level list element is assigned a distance. Non-negated elements get a distance based on their position in the list, where the closer the match is to the start of the list, the shorter the distance is between it and the server. A negated match will be assigned the maximum distance from the server. If there is no match, the address will get a distance which is further than any non-negated list element, and closer than any negated element. For example:

topology {
  10/8;
  !1.2.3/24;
  { 1.2/16; 3/8; };
};

will prefer servers on network 10, followed by hosts on network 1.2.0.0 (netmask 255.255.0.0) and network 3, with the exception of hosts on network 1.2.3 (netmask 255.255.255.0), which is the least preferred.

The default topology is:

topology { localhost; localnets; };

When returning multiple resource records ("RRs"), the name server will normally return them in round robin, that is, after each request, the first RR is put to the end of the list. As the order of RRs is not defined, this should not cause any problems.

The client resolver code should rearrange the RRs as appropriate, for example, using any addresses on the local network before other addresses. However, not all resolvers can do this, or are not correctly configured to do so.

When a client is using a local server, the sorting can be performed by the server, based on the client’s address. This only requires configuring the name servers, not all the clients.

The sortlist statement takes an address match list and interprets it even more specially than the topology statement does.
Each top level statement in the sortlist must itself be an explicit address match list with one or two elements. The first element of each top level list, which may be an IP address, an IP prefix, an acl name or nested address match list, is checked against the source address of the query until a match is found.

Once the source address of the query has been matched, if the top level statement contains only one element, the actual primitive element that matched the source address is used to select the address in the response to move to the beginning of the response. If the statement is a list of two elements, the second element is treated like the address match list in a topology statement. Each top level element is assigned a distance and the address in the response with the minimum distance is moved to the beginning of the response.

In the following example, any queries received from any of the addresses of the host itself will get responses that prefer addresses on any of the locally connected networks. Next most preferred are addresses on the 192.168.1/24 network, and after that either the 192.168.2/24 or 192.168.3/24 network with no preference shown between these two networks. Queries received from a host on the 192.168.1/24 network will prefer other addresses on that network to the 192.168.2/24 and 192.168.3/24 networks. Queries received from a host on the 192.168.4/24 or the 192.168.5/24 network will only prefer other addresses on their directly connected networks.

sortlist {
  { localhost; // IF the local host
    localnets; // THEN first fit on the
    192.168.1/24; // following nets
    { 192.168.2/24; 192.168.3/24; } };
  { 192.168.1/24; // IF on class C 192.168.1
    { 192.168.1/24; // THEN use .1, or .2 or .3
      { 192.168.2/24; 192.168.3/24; } };
    { 192.168.2/24; // IF on class C 192.168.2
      { 192.168.2/24; // THEN use .2, or .1 or .3
        { 192.168.1/24; 192.168.3/24; };
        { 192.168.3/24; // IF on class C 192.168.3
          { 192.168.3/24; // THEN use .3, or .1 or .2
            { 192.168.1/24; 192.168.2/24; };
            { { 192.168.4/24; 192.168.5/24; }; // if .4 or .5,
              // prefer that net
            };
          };
        };
      };
    };
  };
};

The following example will give reasonable behavior for the local host and hosts on directly connected networks. It is similar to the behavior of the address sort in BIND 4.9.x. Responses sent to queries from the local host will favor any of the directly connected networks. Responses sent to queries from any other hosts on a directly connected network will prefer addresses on that same network. Responses to other queries will not be sorted.

sortlist {
  { localhost; localnets; };
  { localnets; };
};
RRset Ordering

When multiple records are returned in an answer it may be useful to configure the order the records are placed into the response. For example the records for a zone might be configured to always be returned in the order they are defined in the zone file. Perhaps you want a random shuffle of the records as they are returned. The rrset-order statement permits you to configure the order of the records in a multiple record response. The default, if no ordering is defined, is a cyclic ordering (round robin).

An order_spec is defined as follows:

```
[ class class_name ][ type type_name ][ name "FQDN" ] order ordering
```

If no class is specified, the default is ANY. If no type is specified, the default is ANY. If no name is specified, the default is "*".

The legal values for ordering are:

- **fixed**: Records are returned in the order they are defined in the zone file.
- **random**: Records are returned in some random order.
- **cyclic**: Records are returned in a round-robin order.

For example:

```
rrset-order {
    class IN type A name "rc.vix.com" order random;
    order cyclic;
};
```

will cause any responses for type A records in class IN that have "rc.vix.com" as a suffix, to always be returned in random order. All other records are returned in cyclic order.

If multiple rrset-order statements appear, they are not combined. The last one applies.

If no rrset-order statement is specified, the following default statement is used:

```
rrset-order { class ANY type ANY name "*" order cyclic ; };
```

Tuning

- **lame-ttl**: Sets the number of seconds to cache a lame server indication. 0 disables caching. The default is 600 (10 minutes). The maximum value is 1800 (30 minutes).

- **max-ncache-ttl**: To reduce network traffic and increase performance, the server store negative answers. max-ncache-ttl is used to set a maximum retention time for these answers in the server in seconds. The default max-ncache-ttl is 10800 seconds (3 hours). max-ncache-ttl cannot exceed the maximum retention time for ordinary (positive) answers (7 days).
and will be silently truncated to 7 days if set to a value which is greater than 7 days.

**min-roots**

The minimum number of root servers that is required for a request for the root servers to be accepted. The default is 2.

```
zone domain_name [(in | hs | hesiod | chaos)] {
  type master;
  file path_name;
  [ check-names (warn | fail | ignore)];
  [ allow-update {address_match_list};]
  [ allow-query {address_match_list};]
  [ allow-transfer {address_match_list};]
  [ dialup yes_or_no;]
  [ notify yes_or_no;]
  [ also-notify {ip_addr; [ip_addr; ...]};]
  [ pubkey number number number string;]
};
```

```
zone domain_name [(in | hs | hesiod | chaos)] {
  type slave | stub ;
  [ file path_name;]
  masters [port ip_port] {ip_addr; [ip_addr; ...]};
  [ check-names (warn | fail | ignore)];
  [ allow-update {address_match_list};]
  [ allow-query {address_match_list};]
  [ allow-transfer {address_match_list};]
  [ transfer-source ip_addr;]
  [ max-transfer-time-in number;]
  [ notify yes_or_no;]
  [ also-notify {ip_addr; [ip_addr; ...]};]
  [ pubkey number number number string;]
};
```

```
zone domain_name [(in | hs | hesiod | chaos)] {
  type forward;
  [ forward (only | first);]
  [ forwarders { [ip_addr; [ip_addr; ...] ];} ;]
  [ check-names (warn | fail | ignore)];
};
```

```
zone "." [(in | hs | hesiod | chaos)] {
  type hint;
  file path_name;
  [ check-names (warn | fail | ignore)];
};
```

**Definition and Usage**

The zone statement is used to define how information about particular DNS zones is managed by the server. There are five different zone types.

**master**

The server has a master copy of the data for the zone and will be able to provide authoritative answers for it.
A slave zone is a replica of a master zone. The masters list specifies one or more IP addresses that the slave contacts to update its copy of the zone. If a port is specified, it then checks to see if the zone is current and makes zone transfers to the port given. If a file is specified, then the replica will be written to the named file. Use of the file clause is highly recommended, since it often speeds server startup and eliminates a needless waste of bandwidth.

A stub zone is like a slave zone, except that it replicates only the NS records of a master zone instead of the entire zone.

A forward zone is used to direct all queries in it to other servers, as described in THE OPTIONS STATEMENT section. The specification of options in such a zone will override any global options declared in the options statement.

If no forwarders clause is present in the zone or an empty list for forwarders is given, then no forwarding will be done for the zone, cancelling the effects of any forwarders in the options statement. Thus if you want to use this type of zone to change only the behavior of the global forward option, and not the servers used, then you also need to respecify the global forwarders.

The initial set of root name servers is specified using a hint zone. When the server starts up, it uses the root hints to find a root name server and get the most recent list of root name servers.

The zone’s name may optionally be followed by a class. If a class is not specified, class in (for "internet"), is assumed. This is correct for the vast majority of cases.

The hesiod class is for an information service from MIT’s Project Athena. It is used to share information about various systems databases, such as users, groups, and printers. More information can be found at ftp://athena-dist.mit.edu/pub/ATHENA/usenix/athena_changes.PS. The keyword hs is a synonym for hesiod.

Another MIT development was CHAOSnet, a LAN protocol created in the mid-1970s. It is still sometimes seen on LISP stations and other hardware in the AI community, and zone data for it can be specified with the chaos class.

<table>
<thead>
<tr>
<th>Classes</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>The zone’s name may optionally be followed by a class. If a class is not specified, class in (for &quot;internet&quot;), is assumed. This is correct for the vast majority of cases.</td>
<td>check-names See the subsection on Name Checking in THE OPTIONS STATEMENT.</td>
</tr>
<tr>
<td>The hesiod class is for an information service from MIT’s Project Athena. It is used to share information about various systems databases, such as users, groups, and printers. More information can be found at ftp://athena-dist.mit.edu/pub/ATHENA/usenix/athena_changes.PS. The keyword hs is a synonym for hesiod.</td>
<td>allow-query See the description of allow-query in the Access Control subsection of THE OPTIONS STATEMENT.</td>
</tr>
</tbody>
</table>
allow-update | Specifies which hosts are allowed to submit dynamic DNS updates to the server. The default is to deny updates from all hosts.

allow-transfer | See the description of allow-transfer in the Access Control subsection of THE OPTIONS STATEMENT.

transfer-source | transfer-source determines which local address will be bound to the TCP connection used to fetch this zone. If not set, it defaults to a system controlled value which will usually be the address of the interface "closest to" the remote end. This address must appear in the remote end's allow-transfer option for this zone if one is specified.

max-transfer-time-in | See the description of max-transfer-time-in in the Zone Transfers subsection of THE OPTIONS STATEMENT.

dialup | See the description of dialup in the Boolean Options subsection of THE OPTIONS STATEMENT.

notify | See the description of notify in the Boolean Options subsection of the THE OPTIONS STATEMENT.

also-notify | also-notify is only meaningful if notify is active for this zone. The set of machines that will receive a DNS NOTIFY message for this zone is made up of all the listed name servers for the zone (other than the primary master), plus any IP addresses specified with also-notify. also-notify is not meaningful for stub zones. The default is the empty list.

forward | forward is only meaningful if the zone has a forwarders list. The only value causes the lookup to fail after trying the forwarders and getting no answer, while first would allow a normal lookup to be tried.

forwarders | The forwarders option in a zone is used to override the list of global forwarders. If it is not specified in a zone of type forward, no forwarding is done for the zone, and the global options are not used.

pubkey | The DNSSEC flags, protocol, and algorithm are specified, as well as a base-64 encoded string representing the key.

**acl** name {
  address_match_list
};
The acl statement creates a named address match list. It gets its name from a primary use of address match lists: Access Control Lists (acls).

An address match list’s name must be defined with acl before it can be used elsewhere. No forward references are allowed.

The following acls are built-in:

- **any** Allows all hosts.
- **none** Denies all hosts.
- **localhosts** Allows the IP addresses of all interfaces on the system.
- **localnets** Allows any host on a network for which the system has an interface.

The key statement defines a key ID which can be used in a server statement to associate with a particular name server a method of authentication that is more rigorous than simple IP address matching. A key ID must be created with the key statement before it can be used in a server definition or an address match list.

The **algorithm_id** is a string that specifies a security/authentication algorithm. The **secret_string** is the secret to be used by the algorithm, and is treated as a base-64 encoded string. If you have a secret_string in your named.conf file, make sure that it is not be readable by anyone beside superuser.

The trusted-keys statement is for use with DNSSEC-style security, originally specified in RFC 2065. DNSSEC is meant to provide three distinct services: key distribution, data origin authentication, and transaction and request authentication.

The contributed section of the ISC BIND distribution includes a dns_signer utility to sign zone data according to the DNSSEC specifications. The utility is provided as-is, without any expressed or implied warranties. The contributed source could be retrieved from the /isc/bind/src/cur/bind-8 directory at ISC’s FTP site, ftp.isc.org.
Each trusted key is associated with a domain name. Its attributes are the non-negative integral flags, protocol, and algorithm, as well as a base-64 encoded string representing the key.

Any number of trusted keys can be specified.

```plaintext
server ip_addr {
    [ bogus yes_or_no; ]
    [ transfers number; ]
    [ transfer-format ( one-answer | many-answers ); ]
    [ keys { key_id [ key_id ... ] }; ]
};
```

**THE SERVER STATEMENT**

**Syntax**

The server statement defines the characteristics to be associated with a remote name server.

If you discover that a server is giving out bad data, marking it as `bogus` will prevent further queries to it. The default value of `bogus` is `no`.

The server supports two zone transfer methods. The first, `one-answer`, uses one DNS message per resource record transferred. The second method, `many-answers` packs as many resource records as possible into a message. `many-answers` is more efficient, but is only understood by BIND 8.1 and patched versions of BIND 4.9.5. You can specify which method to use for a server with the `transfer-format` option. If `transfer-format` is not specified, the `transfer-format` specified by the options statement will be used.

The transfers will be used in a future release of the server to limit the number of concurrent in-bound zone transfers from the specified server. It is checked for syntax but is otherwise ignored.

The `key` clause is used to identify a `key_id` defined by the key statement, to be used for transaction security when talking to the remote server. The `key` statement must come before the server statement that references it.

The `key` statement is intended for future use by the server. It is checked for syntax but is otherwise ignored.

```plaintext
controls {
    [ inet ip_addr
        port ip_port
        allow { address_match_list; ]; ]
    [ unix path_name
        perm number
        owner number
        group number; ]
};
```

**THE CONTROLS STATEMENT**

**Syntax**
The `controls` statement declares control channels to be used by system administrators to affect the operation of the local name server. These control channels are used by the `ndc(1M)` utility to send commands to and retrieve non-DNS results from a name server.

A UNIX control channel is a FIFO in the file system, and access to it is controlled by normal file system permissions. It is created by `in.named(1M)` with the specified file mode bits, user and group owner. See `chmod(1)`. Note that, unlike `chmod`, the mode bits specified for `perm` will normally have a leading 0 so the number is interpreted as octal. Also note that the user and group ownership specified as owner and group must be given as numbers, not names. It is recommended that the permissions be restricted to administrative personnel only, or else any user on the system may be able to manage the local name server.

An inet control channel is a TCP/IP socket accessible to the Internet, created at the specified `ip_port` on the specified `ip_addr`. Modern telnet clients are capable of speaking directly to these sockets, and the control protocol is ARPAnet-style text. It is recommended that 127.0.0.1 be the only `ip_addr` used, and this only if you trust all non-privileged users on the local host to manage your name server.

```
include path_name;
```

The `include` statement inserts the specified file at the point that the `include` statement is encountered. It cannot be used within another statement, though, so a line such as

```
acl internal_hosts { include internal_hosts.acl; };
```

is not allowed.

Use `include` to break the configuration up into easily-managed chunks. For example:

```
include "/etc/security/keys.bind";
include "/etc/acls.bind";
```

could be used at the top of a BIND configuration file in order to include any acl or key information.

Be careful not to use "`#include`," like you would in a C program, because "`#`" is used to start a comment.

**EXAMPLE 1** Simple Configuration File

The simplest configuration file that is still realistically useful is one which simply defines a hint zone that has a full path to the root servers file, for example:

```
zone "." in {
    type hint;
    file "/var/named/root.cache";
};
```
EXAMPLE 1 Simple Configuration File  (Continued)

EXAMPLE 2 Another Example of a Configuration File

Here is a more typical real-world example.

```c
/*
* A simple BIND 8 configuration
*/

logging {
    category lame-servers { null; };
    category cname { null; };
};

options {
    directory "*/var/named";
};

controls {
    inet * port 52 allow { any; }; // a bad idea
    unix "*/var/run/ndc" perm 0600 owner 0 group 0; // the default
};

zone "isc.org" in {
    type master;
    file "master/isc.org";
};

zone "vix.com" in {
    type slave;
    file "slave/vix.com";
    masters { 10.0.0.53; };
};

zone "0.0.127.in-addr.arpa" in {
    type master;
    file "master/127.0.0";
};

zone "." in {
    type hint;
    file "root.cache";
};
```

FILES

/etc/named.conf  The BIND 8 in.named 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>Interface Stability</td>
<td>Standard BIND 8.2.2</td>
</tr>
</tbody>
</table>

SEE ALSO

chmod(1), in.named(1M), named-bootconf(1M), ndc(1M), syslogd(1M), syslog(3C), syslog.conf(4), attributes(5)

ncad_addr(4)

NAME
ncad_addr – Solaris Network Cache and Accelerator (NCA) socket utility library

SYNOPSIS
/usr/lib/ncad_addr.so

DESCRIPTION
ncad_addr.so() is the Solaris Network Cache and Accelerator (NCA) socket utility library. Use this library with a web server to avoid support for the PF_NCA family type socket. The web server can take advantage of NCA functionality.

Interpose the ncad_addr() interfaces before the interfaces in libsocket() by setting the environment variable LD_PRELOAD to ncad_addr.so so that it is preloaded before libsocket.so.1. The ncad_addr.so interfaces will be interposed only if NCA is enabled. See ncakmod(1).

EXAMPLES
EXAMPLE 1 Interposing ncad_addr

Set LD_PRELOAD as follows to interpose the ncad_addr() socket utility library:
LD_PRELOAD=/usr/lib/ncad_addr.so /usr/bin/httpd

FILES
/usr/lib/ncad_addr.so  ncad_addr() socket utility library shared object

ATTRIBUTES
See attributes(5) for 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>Unstable</td>
</tr>
</tbody>
</table>

SEE ALSO
nca(1), ncab2clf(1), ncakmod(1), ncad(1M), socket(3SOCKET), nca.if(4), ncakmod.conf(4), attributes(5)

NOTES
Only applications that utilize the NCA feature, for example web servers, should interpose this library.
nca.if – the NCA configuration file that specifies physical interfaces

/etc/nca/nca.if

Specify the physical interfaces for which the Solaris Network Cache and Accelerator ("NCA") feature will be configured in the nca.if configuration file. List the physical interfaces in the file, one per line. To configure NCA to listen on all physical interfaces present on the system backed by a hostname.\{interface_name\}, then list only an asterisk ("\*") in nca.if.

When the ncaakmod(1) initialization script is invoked during system boot, it will attempt to configure each physical interface specified in the nca.if file by using ncaconfd(1M). Note that there must be an accompanying hostname.\{interface_name\} file and an entry in /etc/hosts for the contents of hostname.\{interface_name\}.

You must reboot in order to implement changes to the nca.if file.

### EXAMPLES

**IA**

**EXAMPLE 1 nca.if on IA**

The following is an example of an nca.if file that would be used on an IA system:

```
iprb1
iprb6
iprb8
```

**SPARC**

**EXAMPLE 2 nca.if on SPARC**

The following is an example of an nca.if file that would be used on a SPARC system:

```
hme2
hme3
hme4
```

**All Platforms**

**EXAMPLE 3 Configuring NCA to Listen on All Physical Interfaces**

The following example shows the contents of an nca.if file that would be used to configure either platform to listen on all physical interfaces present on the system:

```
*
```

### FILES

- `/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.
- `/etc/hosts` Lists all host names associated with the server. Entries in this file must match with entries in `/etc/hostname.\{\}\{0-9\}` for NCA to function.
nca.if(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>SUNWncar</td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Evolving</td>
</tr>
</tbody>
</table>

**SEE ALSO**

nca(1), ncab2clf(1), ncakmod(1), ifconfig(1M), ncakmod.conf(4), ncalogd.conf(4), attributes(5)

*System Administration Guide, Volume 3*
The ncakmod.conf file is used to configure the Solaris Network Cache and Accelerator ("NCA") kernel module. The file contains two fields, key and value.

The status key is used to indicate if the user wants to have NCA turned on as a feature. If the value of status key is enabled, then the NCA kernel module will be pushed on to the specified interfaces. If the value of the status key is disabled, then the NCA kernel module will not be pushed on to any interfaces. The default is disabled.

The httpd_door_path key specifies the path name of the Solaris Door RPC mechanism that will be used to communicate with the http daemon. The default value is /var/run/nca_httpd_1.door.

The ncad_status key is used to indicate if the user wants to enable the ncad(1M) daemon. The ncad daemon provides a door server for your web server if you do not have that capability built in but want to use the NCA feature. The default value of ncad_status is disabled. If the value of ncad_status is enabled, then the user daemon will be started at boot time.

Use the nca_active key to indicate whether to allow NCA to actively open outgoing TCP connections. The default value for nca_active is disabled. If set to enabled, ncacnf sets up NCA for each interface and then operates as a daemon, allowing NCA to make outgoing TCP connections. This functionality is possible only by using the doors interface to NCA. A web server that uses the sockets interface with PF_NCA or ncad_addr.so cannot connect by means of nca_active.

In order to implement changes to the ncakmod.conf file, you will need to reboot.

**EXAMPLE 1** A Sample ncakmod.conf File

The following is a sample ncakmod.conf file:

```
# NCA Kernel Module Configuration File
# status-disabled
httpd_door_path=/var/run/nca_httpd_1.door
ncad_status-disabled
nca_active-disabled
```

**FILES**

/etc/nca/ncakmod.conf

The NCA kernel module configuration 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>SUNWncar</td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Evolving</td>
</tr>
</tbody>
</table>

SEE ALSO nca(1), ncab2clf(1), ncakmod(1), ncad(1M), door_create(3DOOR), nca.if(4), ncad_addr(4), ncalogd.conf(4), attributes(5)

*System Administration Guide, Volume 3*
The ncalgd.conf is used to configure Solaris Network Cache and Accelerator (“NCA”) logging. The file contains two fields, key and value.

The status key is used to indicate if the user wants to have NCA logging turned on. If the value of status key is enabled, then NCA logging will be turned on. If the value of the status key is disabled, then NCA logging will not be invoked. The default value is disabled.

The logd_path_name key specifies the location of the log file. The value of logd_path_name is the absolute path to the log file. The default value is /var/nca/log. logd_path_name can also contain a white space delimited list of values for multiple log files to a maximum of 16. NCA logging moves to the next file on the list once the file size specified by logd_file_size has been reached. When the last file is full, NCA logging rotates back to the first file in the list. A pointer to the current log file is stored in /var/nca/current.

The logd_file_size key specifies the value of the file size, in bytes, allowed for each log file specified in by the logd_path_name key. The default value is 1000000 bytes.

In order to implement changes to the ncalgd.conf file, you will need to stop and start NCA logging or reboot.

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.

Examples

Example 1 A Sample ncalgd.conf File

The following is a sample ncalgd.conf file that specifies three log files:

```
# NCA Log Daemon Configuration File

status=disabled
logd_path_name=/var/nca/log1 /var/nca/log2 /var/nca/log3
logd_file_size=1000000
```

Note that there is no NCA logging daemon. Logging is performed as one of the functions of the NCA software.
See attributes(5) for 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), ncakmod(1), door_create(3X), nca.if(4), ncakmod.conf(4), attributes(5)

System Administration Guide, Volume 3
The ndpd.conf file contains configuration information for in.ndpd(1M) when used on a router. This file does not need to exist or can be empty on a host. The file has one configuration entry per line; note that lines can be extended with “\” followed by a newline. There are four forms of configuration entries which are identified by the first field on the line: ifdefault, prefixdefault, if, or prefix. The ifdefault and if entries set interface configuration variables; the former establishes the defaults for all interfaces. Any ifdefault entries must precede any if entries in the file.

The prefixdefault and prefix entries control per-prefix configuration variables. prefixdefault establishes the defaults for all prefixes on all interfaces. Any prefixdefault entries must precede any prefix entries in the file.

Each ifdefault entry is composed of a single line of the form:

ifdefault [ if-variable-name value ]*

Each if entry is composed of a single line of the form:

if interface [ if-variable-name value ]*

Each prefixdefault entry is composed of a single line of the form:

prefixdefault [ prefix-variable-name value ]*

Each prefix entry is composed of a single line of the form:

prefix prefix/prefix_length interface [ prefix-variable-name value ]*

Fields are separated by either SPACE or TAB characters. A ‘#’ (number sign) indicates the beginning of a comment. Characters up to the end of the line are not interpreted by routines that search this file.

interface The name of a network interface, for example, le0.
prefix An IPv6 address in standard hexadecimal notation, for example, fec0:0:0:1::0.
prefix_length A number between 0 and 128.
if-variable-name An interface variable as discussed in RFC 2461 and RFC 2462. The following lists the each interface variable and its default value and unit:
A prefix variable as discussed in RFC 2461 and RFC 2462. The following lists each interface variable and its default value and unit:

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Default</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdvValidLifetime</td>
<td>2592000</td>
<td>Seconds</td>
</tr>
<tr>
<td>AdvOnLinkFlag</td>
<td>true</td>
<td>Boolean</td>
</tr>
<tr>
<td>AdvPreferredLifetime</td>
<td>604800</td>
<td>Seconds</td>
</tr>
<tr>
<td>AdvAutonomousFlag</td>
<td>true</td>
<td>Boolean</td>
</tr>
<tr>
<td>AdvValidExpiration</td>
<td>not set</td>
<td>Date/Time</td>
</tr>
<tr>
<td>AdvPreferredExpiration</td>
<td>not set</td>
<td>Date/Time</td>
</tr>
</tbody>
</table>

The "Expiration" variables are used to specify that the lifetime should be decremented in real time as specified in RFC 2461. If an "Expiration" variable is set then it takes precedence over the corresponding "Lifetime" variable setting.

The value is a function of the unit. Boolean values are true, false, on, off, 1, or 0.
Values in seconds can have characters appended for day (d), hour (h), minute (m) and second (s). The default is seconds. For example, 1h means 1 hour. This is equivalent to the value 3600.

Values in milliseconds can have characters appended for day (d), hour (h), minute (m) second (s), and millisecond (ms). The default is milliseconds. For example, 1h is equivalent to the value 3600000.

Date/time values are strings that use the recommended ISO date format described as "%Y-%m-%d %R", which represents a 4 digit year, a dash character, a numeric month, a dash character, and a numeric day of the month, followed by one or more whitespace characters and finally a 24 hour clock with hours, a colon, and minutes. For example, 1999-01-31 20:00 means 8pm January 31 in 1999. Since the date/time values contain a space, use single or double quotes to declare the value. For example:

```
prefixdefault AdvPreferredExpiration '1999-01-31 20:00'
```

EXAMPLES

EXAMPLE 1 Sending Router Advertisements for all Interfaces

The following example can be used to send router advertisements out to all interfaces:

```bash
# Send router advertisements out all interfaces
ifdefault AdvSendAdvertisements on
prefixdefault AdvOnLinkFlag on AdvAutonomousFlag on

# Advertise a (bogus) global prefix and a site
# local prefix on three interfaces using the default lifetimes
prefix 2:0:0:9255::/64 hme0
prefix fec0:0:0:9255::/64 hme0
prefix 2:0:0:9265::/64 hme1
prefix fec0:0:0:9265::/64 hme1
prefix 2:0:0:9259::/64 hme2
prefix fec0:0:0:9259::/64 hme2
```

ATTRIBUTES

See attributes(5) for 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>
</tbody>
</table>

SEE ALSO

in.ndpd(1M),attributes(5), icmp6(7P), ip6(7P)

netconfig – network configuration database

/etc/netconfig

The network configuration database, /etc/netconfig, is a system file used to store information about networks that are connected to the system. The netconfig database and the routines that access it (see getnetconfig(3NSL)) are part of the Network Selection component. The Network Selection component also includes getnetpath(3NSL) routines to provide application-specific network search paths. These routines access the netconfig database based on the environment variable NETPATH. See environ(5).

netconfig contains an entry for each network available on the system. Entries are separated by newlines. Fields are separated by whitespace and occur in the order in which they are described below. Whitespace can be embedded as "\blank" or "\tab". Backslashes may be embedded as "\". Lines in /etc/netconfig that begin with a # (hash) in column 1 are treated as comments.

Each of the valid lines in the netconfig database correspond to an available transport. Each entry is of the form:

```
network ID  semantics  flag  protocol-family  protocol-name  
            network-device  translation-libraries
```

**network ID**
A string used to uniquely identify a network. network ID consists of non-null characters, and has a length of at least 1. No maximum length is specified. This namespace is locally significant and the local system administrator is the naming authority. All network IDs on a system must be unique.

**semantics**
The semantics field is a string identifying the “semantics” of the network, that is, the set of services it supports, by identifying the service interface it provides. The semantics field is mandatory. The following semantics are recognized.

- `tpi_clts`: Transport Provider Interface, connectionless
- `tpi_cots`: Transport Provider Interface, connection oriented
- `tpi_cots_ord`: Transport Provider Interface, connection oriented, supports orderly release

**flag**
The flag field records certain two-valued (“true” and “false”) attributes of networks. flag is a string composed of a combination of characters, each of which indicates the value of the corresponding attribute. If the character is present, the attribute is
“true.” If the character is absent, the attribute is “false.” “-” indicates that none of the attributes are present. Only one character is currently recognized:

v Visible (“default”) network. Used when the environment variable NETPATH is unset.

protocol family

The protocol family and protocol name fields are provided for protocol-specific applications. The protocol family field contains a string that identifies a protocol family. The protocol family identifier follows the same rules as those for network IDs; the string consists of non-null characters, it has a length of at least 1, and there is no maximum length specified. A “-” in the protocol family field indicates that no protocol family identifier applies (the network is experimental). The following are examples:

loopback Loopback (local to host).
inet Internetwork: UDP, TCP, and the like.
inet6 Internetwork over IPv6: UDP, TCP, and the like.
implink ARPANET imp addresses
pup PUP protocols: for example, BSP
chaos MIT CHAOS protocols
ns XEROX NS protocols
nbs NBS protocols
ecma European Computer Manufacturers Association
datakit DATAKIT protocols
ccitt CCITT protocols, X.25, and the like.
sna IBM SNA
decnet DECNET
dli Direct data link interface
lat LAT
hylink NSC Hyperchannel
appletalk Apple Talk
Network Interface Tap

IEEE 802; also ISO 8802

Umbrella for all families used by OSI (for example, protosw lookup)

CCITT X.25 in particular

AFI = 47, IDI = 4

U.S. Government OSI

The protocol name field contains a string that identifies a protocol. The protocol name identifier follows the same rules as those for network IDs; that is, the string consists of non-NULL characters, it has a length of at least 1, and there is no maximum length specified. A "−" indicates that none of the names listed apply. The following protocol names are recognized.

tcp Transmission Control Protocol

udp User Datagram Protocol

icmp Internet Control Message Protocol

The network device is the full pathname of the device used to connect to the transport provider. Typically, this device will be in the /dev directory. The network device must be specified.

The name-to-address translation libraries support a "directory service" (a name-to-address mapping service) for the network. A "−" in this field indicates the absence of any translation libraries. This has a special meaning for networks of the protocol family inet: its name-to-address mapping is provided by the name service switch based on the entries for hosts and services in nsswitch.conf. For networks of other families, a "−" indicates non-functional name-to-address mapping. Otherwise, this field consists of a comma-separated list of pathnames to dynamically linked libraries. The pathname of the library can be either absolute or relative. See dlopen(3DL).

Each field corresponds to an element in the struct netconfig structure. The identifiers described on this manual page are defined in <netconfig.h>. This structure includes the following members:

char *nc_netid

Network ID, including NULL terminator.
The `nc_semantics` field takes the following values, corresponding to the semantics identified above:

- `NC_TPI_CLTS`
- `NC_TPI_COTS`
- `NC_TPI_COTS_ORD`

The `nc_flag` field is a bitfield. The following bit, corresponding to the attribute identified above, is currently recognized. `NC_NOFLAG` indicates the absence of any attributes.

`NC_VISIBLE`

EXAMPLE 1 A Sample `netconfig` File

Below is a sample `netconfig` file:

```bash
# The "Network Configuration" File.
# Each entry is of the form:
# <networkid> <semantics> <flags> <protofamily> <protoname><device> \ 
# <nametoaddrlibs>
#
# The "." in <nametoaddrlibs> for inet family transports indicates
# redirection to the name service switch policies for "hosts" and
# "services". The "-" may be replaced by nametoaddr libraries that
# comply with the SVr4 specs, in which case the name service switch
```
EXAMPLE 1 A Sample netconfig File

# will not be used for netdirgetbyname, netdirgetbyaddr,
# gethostbyname, gethostbyaddr, getservbyname, and getservbyport.
# There are no nametoaddrlibs for the inet family in Solaris anymore.
#
# The following two entries starting with udp6 and tcp6 are meant to be
# used for IPv6. If you have IPv6 enabled on your machine then you can
# uncomment these two lines to enable RPC and NFS to use the IPv6 stack.
# Consult your network administrator before uncommenting.
#
#udp6  tpi_clts  v  inet6  udp  /dev/udp6  -
tcp6  tpi_cots_ord  v  inet6  tcp  /dev/tcp6  -

udp  tpiclts  v  inet  udp  /dev/udp  -
tcp  tpicotsord  v  inet  tcp  /dev/tcp  -
rawip  tpiraw  -  inet  -  /dev/rawip  -
ticlts  tpiclts  v  loopback  -  /dev/ticlts  straddr.so
ticotsord  tpicotsord  v  loopback  -  /dev/ticotsord  straddr.so
ticots  tpicots  v  loopback  -  /dev/ticots  straddr.so

FILES

<netconfig.h>

SEE ALSO
dlopen(3DL), getnetconfig(3NSL), getnetpath(3NSL), nsswitch.conf(4)

System Administration Guide, Volume 3

Network Interface Guide
A netgroup defines a network-wide group of hosts and users. Netgroups may be used to restrict access to shared NFS filesystems and for restricting remote login and shell access.

Network groups are stored in one of the Network Information Services, either NIS or NIS+, not in a local file.

This manual page describes the format for a file that may be used to supply input to the makedbm(1M) or nisaddent(1M) programs that are use to build the NIS map or NIS+ table, respectively.

Each line of the file defines the name and membership of network group. The line should have the format:

```
groupname  member . . .
```

The items on a line may be separated by a combination of one or more spaces or tabs.

The `groupname` is the name of the group being defined. This is followed by a list of members of the group. Each `member` is either another group name, all of whose members are to be included in the group being defined, or a triple of the form:

```
(hostname,username,domainname)
```

In each triple, any of the three fields `hostname`, `username`, and `domainname`, can be empty. An empty field signifies a "wildcard" matching any value in that field. Thus:

```
everything ( , ,this.domain)
```

defines a group named "everything" for the domain "this.domain" to which every host and user belongs.

The `domainname` field refers to the domain in which the triple is valid, not the domain containing the host or user.

Netgroups can be used to control NFS mount access (see share_nfs(1M)) and to control remote login and shell access (see hosts.equiv(4)). They can also be used to control local login access (see passwd(4), shadow(4), and "compat" in nsswitch.conf(4)).

When used for these purposes, a host is considered a member of a netgroup if the netgroup contains any triple in which the `hostname` field matches the name of the host requesting access and the `domainname` field matches the domain of the host controlling access.
Similarly, a user is considered a member of a netgroup if the netgroup contains any triple in which the **username** field matches the name of the user requesting access and the **domainname** field matches the domain of the host controlling access.

Note that when netgroups are used to control NFS mount access, access is granted depending only on whether the requesting host is a member of the netgroup. Remote login and shell access can be controlled both on the basis of host and user membership in separate netgroups.

/\etc/netgroup \used by /\var/yp/Makefile on NIS masters to build the NIS netgroup map

Note that the netgroup information must always be stored in a network information service, either NIS or NIS+. The local file is only used to construct the netgroup NIS maps or NIS+ table; it is never consulted directly.

**SEE ALSO**
- nis+(1), makedbm(1M), nisaddent(1M), share_nfs(1M), innetgr(3C), hosts(4), hosts.equiv(4), nsswitch.conf(4), passwd(4), shadow(4)
- netgroup requires NIS or NIS+.

Applications may make general membership tests using the `innetgr()` function (see `innetgr(3C)`).

Because the "-" character will not match any specific username or hostname, it is commonly used as a placeholder that will match only wildcarded membership queries. So, for example:

```plaintext
onlyhosts  (host1,-,our.domain) (host2,-,our.domain)
onlyusers  (-,john,our.domain) (-,linda,our.domain)
```

effectively define netgroups containing only hosts and only users, respectively. Any other string that is guaranteed not to be a legal username or hostname will also suffice for this purpose.

Use of placeholders will improve search performance.

When a machine with multiple interfaces and multiple names is defined as a member of a netgroup, one must list all of the names (see `hosts(4)`). A manageable way to do this is to define a netgroup containing all of the machine names. For example, for a host "gateway" that has names "gateway-subnet1" and "gateway-subnet2" one may define the netgroup:

```plaintext
gateway (gateway-subnet1, ,our.domain) (gateway-subnet2, ,our.domain)
```

and use this netgroup gateway whenever the host is to be included in another netgroup.
netid(4)

NAME
netid – netname database

SYNOPSIS
/etc/netid

DESCRIPTION
The netid file is a local source of information on mappings between netnames (see
secure_rpc(3NSL)) and user ids or hostnames in the local domain. The netid file
can be used in conjunction with, or instead of, the network source: NIS or NIS+.
The publickey entry in the nsswitch.conf (see nsswitch.conf(4)) file determines
which of these sources will be queried by the system to translate netnames to local
user ids or hostnames.

Each entry in the netid file is a single line of the form:

netname uid : gid, gid, gid ...

or

netname 0 : hostname

The first entry associates a local user id with a netname. The second entry associates a
hostname with a netname.

The netid file field descriptions are as follows:

netname
The operating system independent network name for the user or
host. netname has one of two formats. The format used to specify a
host is of the form:

unix.hostname@domain

where hostname is the name of the host and domain is the
network domain name.

The format used to specify a user id is of the form:

unix.uid@domain

where uid is the numerical id of the user and domain is the network
domain name.

uid
The numerical id of the user (see passwd(4)). When specifying a
host name, uid is always zero.

group
The numerical id of the group the user belongs to (see group(4)).
Several groups, separated by commas, may be listed for a single
uid.

hostname
The local hostname (see hosts(4)).

Blank lines are ignored. Any part of a line to the right of a ‘#’ symbol is treated as a
comment.
EXAMPLE 1 A sample netid file

Here is a sample netid file:

unix.789@West.Sun.COM 789:30,65
unix.123@Bldg_xy.Sun.COM 123:20,1521
unix.candlestick@campus1.bayarea.EDU 0:candlestick

FILES
/etc/group        groups file
/etc/hosts        hosts database
/etc/netid        netname database
/etc/passwd       password file
/etc/publickey    public key database

SEE ALSO
netname2user(3NSL), secure_rpc(3NSL), group(4), hosts(4),
nsswitch.conf(4), passwd(4), publickey(4)
The netmasks file contains network masks used to implement IP subnetting. It supports both standard subnetting as specified in RFC-950 and variable length subnetting as specified in RFC-1519. When using standard subnetting there should be a single line for each network that is subnetted in this file with the network number, any number of SPACE or TAB characters, and the network mask to use on that network. Network numbers and masks may be specified in the conventional IP ‘.’ (dot) notation (like IP host addresses, but with zeroes for the host part). For example,

```
128.32.0.0 255.255.255.0
```

can be used to specify that the Class B network 128.32.0.0 should have eight bits of subnet field and eight bits of host field, in addition to the standard sixteen bits in the network field.

When using variable length subnetting, the format is identical. However, there should be a line for each subnet with the first field being the subnet and the second field being the netmask that applies to that subnet. The users of the database, such as `ifconfig(1M)`, perform a lookup to find the longest possible matching mask. It is possible to combine the RFC-950 and RFC-1519 form of subnet masks in the netmasks file. For example,

```
128.32.0.0 255.255.255.0
128.32.27.0 255.255.255.240
128.32.27.16 255.255.255.240
128.32.27.32 255.255.255.240
128.32.27.48 255.255.255.240
128.32.27.64 255.255.255.240
128.32.27.80 255.255.255.240
128.32.27.96 255.255.255.240
128.32.27.112 255.255.255.240
128.32.27.128 255.255.255.240
128.32.27.144 255.255.255.240
128.32.27.160 255.255.255.240
128.32.27.176 255.255.255.240
128.32.27.192 255.255.255.240
128.32.27.208 255.255.255.240
128.32.27.224 255.255.255.240
128.32.27.240 255.255.255.240
128.32.64.0 255.255.255.192
```

can be used to specify different netmasks in different parts of the 128.32.0.0 Class B network number. Addresses 128.32.27.0 through 128.32.27.255 have a subnet mask with 28 bits in the combined network and subnet fields (often referred to as the subnet field) and 4 bits in the host field. Furthermore, addresses 128.32.64.0 through 128.32.64.63 have a 26 bits in the subnet field. Finally, all other addresses in the range 128.32.0.0 through 128.32.255.255 have a 24 bit subnet field.
Invalid entries are ignored.

**SEE ALSO**

ifconfig(1M), inet(7P)


**NOTES**

/etc/inet/netmasks is the official SVr4 name of the netmasks file. The symbolic link /etc/netmasks exists for BSD compatibility.
The .netrc file contains data for logging in to a remote host over the network for file transfers by ftp(1). This file resides in the user’s home directory on the machine initiating the file transfer. Its permissions should be set to disallow read access by group and others (see chmod(1)).

The following tokens are recognized; they may be separated by SPACE, TAB, or NEWLINE characters:

- **machine name**: Identify a remote machine name. The auto-login process searches the .netrc file for a machine token that matches the remote machine specified on the ftp command line or as an open command argument. Once a match is made, the subsequent .netrc tokens are processed, stopping when the EOF is reached or another machine token is encountered.

- **login name**: Identify a user on the remote machine. If this token is present, the auto-login process will initiate a login using the specified name.

- **password string**: Supply a password. If this token is present, the auto-login process will supply the specified string if the remote server requires a password as part of the login process. Note: if this token is present in the .netrc file, ftp will abort the auto-login process if the .netrc is readable by anyone besides the user.

- **account string**: Supply an additional account password. If this token is present, the auto-login process will supply the specified string if the remote server requires an additional account password, or the auto-login process will initiate an ACCT command if it does not.

- **macdef name**: Define a macro. This token functions the same as ftp macdef. A macro is defined with the specified name; its contents begin with the next .netrc line and continue until a null line (consecutive NEWLINE characters) is encountered. If a macro named init is defined, it is automatically executed as the last step in the auto-login process.

**EXAMPLE 1** A Sample .netrc File

A .netrc file containing the following line:

```
machine ray login demo password mypassword
```

allows an autologin to the machine ray using the login name demo with password mypassword.

**FILES**

```
~/.netrc
```

**SEE ALSO**

chmod(1), ftp(1), in.ftpd(1M)
networks file is a local source of information regarding the networks which comprise the Internet. The networks file can be used in conjunction with, or instead of, other networks sources, including the NIS maps networksbyname and networksbyaddr and the NIS+ table networks. Programs use the getnetbyname(3SOCKET) routines to access this information.

The network file has a single line for each network, with the following information:

- **official-network-name**
- **network-number**
- **aliases**

Items are separated by any number of SPACE and/or TAB characters. A ‘#’ indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file. This file is normally created from the official network database maintained at the Network Information Control Center (NIC), though local changes may be required to bring it up to date regarding unofficial aliases and/or unknown networks.

Network numbers may be specified in the conventional dot (’.’) notation using the inet_network routine from the Internet address manipulation library, inet(7P). Network names may contain any printable character other than a field delimiter, NEWLINE, or comment character.

**SEE ALSO**

getnetbyaddr(3SOCKET), getnetbyname(3SOCKET), inet(3SOCKET), nsswitch.conf(4), inet(7P)

**NOTES**

The official SVR4 name of the networks file is /etc/inet/networks. The symbolic link /etc/networks exists for BSD compatibility.

The network database does not support subnet masks in general, so getnetbyaddr(3SOCKET) cannot differentiate between networks of 11.128.0.0/255.192.0.0 and 11.128.0.0/255.240.0.0.
nfslog.conf – NFS server logging configuration file

/etc/nfs/nfslog.conf

The nfslog.conf file specifies the location of the NFS server logs, as well as the location of the private work files used by the NFS server and nfslogd(1M) daemon during logging. Each entry in the file consists of a mandatory tag identifier and one or more parameter identifiers. The parameter identifier specifies the value or location of the specific parameter. For instance, the parameter identifier "log=/var/nfs/logs/serverLog" specifies the location of the NFS server activity log. The mandatory tag identifier serves as an index into the /etc/nfs/nfslog.conf file to identify the various parameters to be used. At export time, the share_nfs(1M) command specifies the NFS server logging parameters to use by associating a tag from the /etc/nfs/nfslog.conf file to the exported file system. It is legal for more than one file system to be exported using the same logging tag identifier.

A "global" tag identifier is included in /etc/nfs/nfslog.conf. It specifies the default set of values to be used during logging. If no tag identifier is specified at export time, then the values in the "global" entry are used. The "global" values can be modified by updating this entry in /etc/nfs/nfslog.conf.

Each entry in the file must contain a mandatory tag identifier and at least one parameter/value pair. If a parameter is not specified in a given entry, the global value of the parameter will be used. The exact entry syntax follows:

```plaintext
<tag> [defaultdir=<path>] [log=<path><file>] [fhtable=<path><file>] \\
    [buffer=<path><file>] [logformat=basic|extended]
```

- `defaultdir=<path>` Specifies the directory where the logging files and working files will be placed. This path is prepended to all relative paths specified in other parameters.
- `log=<path><file>` Specifies the location of the user-readable log file. The log will be located in the defaultdir, unless `<path>` is an absolute path.
- `fhtable=<path><file>` Specifies the location of the private file handle to path mapping database files. These database files are for the private use of the NFS server kernel module and the nfslogd daemon. These files will be located in the defaultdir, unless `<path>` is an absolute path. These database files are permanently stored in the file system. Consult nfslogd(1M) for information on pruning the database files.
buffer=<path><file>

Specifies the location of the private work buffer file used by the NFS server kernel module to record raw RPC information. This file is later processed by the nfslog daemon, which in turn generates the user-readable log file. This work buffer file will be located in the defaultdir, unless <path> is an absolute path.

logformat=basic|extended

Sets the format of the user-readable log file. If not specified, the basic format is used. The basic format is compatible with log files generated by the Washington University FTPd. The extended format provides a more detailed log, which includes directory modification operations not included in the basic format, such as mkdir, rmdir and remove. Note that the extended format is not compatible with Washington University’s FTPd log format.

EXAMPLE 1 Using the global Tag

The "global" tag may be modified so that all exported file systems that enabled logging use a common set of parameters that conform to the specific needs of the user. These values are used until a specific tag identifier overrides them.

global defaultdir=/var/nfs log=logs/nfslog \ 
  fhtable=tables/fhtable buffer=buffers/nfslog_workbuffer \ 
  logformat=basic

EXAMPLE 2 Overriding the Global defaultdir and logformat

Because log files can become very large, it may be desirable to store the logs and working files in separate file systems. This can be easily accomplished by simply specifying a different defaultdir for every file system exported by means of a unique tag:

engineering defaultdir=/engineering/logging \ 
  logformat=extended
accounting defaultdir=/accounting/logging
marketing defaultdir=/marketing/logging

File systems shared with the engineering identifier will have their logs and workfiles located in /engineering/logging. For instance, the log file will be located at /engineering/logging/logs/nfslog. Note that the engineering log file will be stored in the extended format, while the rest of the log files will remain in the basic format.

Any of the parameters can be updated in a tag identifier, which overrides the global settings.
nfslog.conf(4)

ATTRIBUTES

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

SEE ALSO

nfslogd(1M), share_nfs(1M), attributes(5)

NOTES

Logs, work files, and file handle to path mapping database can become very large. Be aware of appropriate placement within the file system name space. See nfslogd(1M)) for information on pruning the database files and cycling logs.
DESCRIPTION

The Network Information Service Plus (NIS+) uses a memory based, replicated database. This database uses a set of files in the /var/nis directory for checkpointing to table storage and for maintaining a transaction log. Additionally, the NIS+ server and client use files in this directory to store binding and state information.

The NIS+ service implements an authentication and authorization system that is built upon Secure RPC. In this implementation, the service uses a table named `cred.org_dir.domain-name` to store the public and private keys of principals that are authorized to access the NIS+ namespace. It stores group access information in the subdomain `groups_dir.domain-name` as `group` objects. These two tables appear as files in the /var/nis/data directory on the NIS+ server.

Unlike the previous versions of the network information service, in NIS+, the information in the tables is initially loaded into the service from the ASCII files on the server and then updated using NIS+ utilities (see `nistbladm(1)`). Some sites may wish to periodically regenerate the ASCII files for archival purposes. To do this, a script should be added in the `crontab(1)` of the server that lists these tables and creates the ASCII file from the result.

Note that except for the `NIS_COLDSTART` and `NIS_SHARED_DIRCACHE` file, no other files should be manipulated by commands such as `cp(1)`, `mv(1)` or `rm(1)`. The transaction log file keeps logs of all changes made, and hence the files cannot be manipulated independently.

The files described below are stored in the /var/nis directory:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIS_COLDSTART</td>
<td>Contains NIS+ directory objects that are to be preloaded into the NIS+ cache at startup time. This file is usually created at NIS+ installation time. See <code>nisinit(1M)</code> or <code>nisclient(1M)</code>.</td>
</tr>
<tr>
<td>NIS_SHARED_DIRCACHE</td>
<td>Contains the current cache of NIS+ bindings being maintained by the cache manager. The contents can be viewed with <code>nisshowcache(1M)</code>.</td>
</tr>
<tr>
<td>client_info</td>
<td>Contains configuration information (preferred servers, options, etc.) for <code>nis_cachemgr(1M)</code> and (potentially) other NIS+ clients on the system. It is manipulated by the <code>nisprefadm(1M)</code> command.</td>
</tr>
<tr>
<td>.pref_servers</td>
<td>A cached copy of preferred server information. It is maintained by <code>nis_cachemgr</code>. Do not edit this file manually.</td>
</tr>
<tr>
<td>trans.log</td>
<td>Contains a transaction log that is maintained by the NIS+ service. It can be viewed using the <code>nislog(1M)</code> command. This file contains holes. Its apparent size is...</td>
</tr>
</tbody>
</table>
may be a lot higher than its actual size. There is only one transaction log per server.

| **data.dict** | A dictionary that is used by the NIS+ database to locate its files. It is created by the default NIS+ database package. |
| **data.dict.log** | The log file for the database dictionary. When the server is checkpointed (see the -C option of nisping(1M)), this file will be deleted. |
| **data** | Contains databases that the server uses. |
| **data/root.object** | On root servers, this file contains a directory object that describes the root of the name space. |
| **data/parent.object** | On root servers, this file contains a directory object that describes the parent namespace. This file is created by the nisinit(1M) command. |
| **data/table_name** | For each table in the directory there is a file with the same name that stores the information about that table. If there are subdirectories within this directory, the database for the table is stored in the file, `table_name.subdirectory`. |
| **data/table_name.log** | Contains the database log for the table `table_name`. The log file maintains the state of individual transactions to each database. When a database has been checkpointed (that is, all changes have been made to the `data/table_name` stable storage), this log file will be deleted. Currently, NIS+ does not automatically do checkpointing. The system administrator may want to do nisping -C operations periodically (such as, once a day) to checkpoint the log file. This can be done either through a cron(1M) job, or manually. |
| **data/root_dir** | On root servers, this file stores the database associated with the root directory. It is similar to other table databases. The corresponding log file is called `root_dir.log`. |
| **data/cred.org_dir** | Table containing the credentials of principals in this NIS+ domain. |
| **data/groups_dir** | Table containing the group authorization objects needed by NIS+ to authorize group access. |
| **data/serving_list** | Contains a list of all NIS+ directories that are being served by the NIS+ server on this server. When this
server is added or deleted from any NIS+ directory object, this file is updated by the server.

SEE ALSO

\texttt{cp(1), crontab(1), mv(1), nis(1), nis\_cachemgr(1M), niscat(1), nismatch(1), nistbladm(1), rm(1), cron(1M), nisclient(1M), nisinit(1M), nislog(1M), nisping(1M), nisprefadm(1M), nisshowcache(1M), nis\_objects(3NSL)}
nodename(4)

NAME  nodename – local source for system name

SYNOPSIS  /etc/nodename

DESCRIPTION  When a machine is standalone or its IP address is configured locally, the /etc/nodename file contains the system name. By convention, the system name is the same as the hostname associated with the IP address of the primary network interface, for example, hostname.hme0.

If the machine’s network configuration is managed remotely and delivered by the DHCP or RPC bootparams protocols, the /etc/nodename file is not used, as the system name is delivered by the remote service.

Given a system name value, regardless of source, the uname utility invoked with the -S option is used to set the system name of the running system.

EXAMPLES

EXAMPLE 1 Syntax

The syntax for nodename consists of a single line containing the system’s name. For example, for a system named myhost:

myhost

ATTRIBUTES  See attributes(5) for 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  nis+(1), uname(1), named(1M), ypbind(1M), attributes(5)

NOTES  The nodename file is modified by Solaris installation and de-installation scripts. The user should not edit the file.
<table>
<thead>
<tr>
<th>NAME</th>
<th>nologin – message displayed to users attempting to log on in the process of a system shutdown</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNOPSIS</td>
<td>/etc/nologin</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>The /etc/nologin file contains the message displayed to users attempting to log on to a machine in the process of being shutdown. After displaying the contents of the nologin file, the login procedure terminates, preventing the user from logging onto the machine. This procedure is preferable to terminating a user’s session by shutdown shortly after the user has logged on. Logins by super-user are not affected by this procedure. The message contained in the nologin file is editable by super-user. A typical nologin file contains a message similar to: NO LOGINS: System going down in 10 minutes.</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>login(1), rlogin(1), telnet(1), shutdown(1M)</td>
</tr>
<tr>
<td>NAME</td>
<td>note – specify legal annotations</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>SYNOPSIS</td>
<td>/usr/lib/note</td>
</tr>
</tbody>
</table>
| DESCRIPTION| Each file in this directory contains the NOTE (also _NOTE) annotations legal for a single tool. The name of the file, by convention, should be the tool vendor's stock name, followed by a hyphen, followed by the tool name. For example, for Sun's lock_lint tool the filename should be SUNW-lock_lint. The file should contain the names of the annotations understood by the tool, one per line. For example, if a tool understands the following annotations:

```
NOTE(NOT_REACHED)
NOTE(MUTEX_PROTECTS_DATA(list_lock, list_head))
```

then its file in /usr/lib/note should contain the entries:

```
NOT_REACHED
MUTEX_PROTECTS_DATA
```

Blank lines, and lines beginning with a pound (#), are ignored.

While /usr/lib/note is the default directory tools search for such files, they can be made to search other directories instead simply by setting environment variable NOTEPATH to contain the paths, separated by colons, of directories to be searched, e.g., /usr/mytool/note:/usr/lib/note.

<table>
<thead>
<tr>
<th>USAGE</th>
<th>These files are used by such tools whenever they encounter NOTES they do not understand. If a file in /usr/lib/note contains the annotation, then it is valid. If no such file contains the annotation, then the tool should issue a warning complaining that it might be invalid.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENVIRONMENT VARIABLES</td>
<td>NOTEPATH specify paths to be searched for annotation files. Paths are separated by colons (&quot;;&quot;).</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>NOTE(3EXT)</td>
</tr>
</tbody>
</table>
NAME | nscd.conf – name service cache daemon configuration
SYNOPSIS | /etc/nscd.conf
DESCRIPTION | The nscd.conf file contains the configuration information for nscd(1M). Each line specifies either an attribute and a value, or an attribute, cachename, and a value. Fields are separated either by SPACE or TAB characters. A ‘#’ (number sign) indicates the beginning of a comment; characters up to the end of the line are not interpreted by nscd.

cachename is represented by hosts, ipnodes, passwd, or groups.

attribute supports the following:

logfile debug-file-name
Specifies name of the file to which debug info should be written. Use /dev/tty for standard output.

debug-level value
Sets the debug level desired. value may range from 0 (the default) to 10. Use of this option causes nscd(1M) to run in the foreground and not become a daemon. Note that the output of the debugging command is not likely to remain the same from release-to-release; scripts should not rely on its format.

enable-cache cachename value
Enables or disables the specified cache. value may be either yes or no.

positive-time-to-live cachename value
Sets the time-to-live for positive entries (successful queries) in the specified cache. value is in integer seconds. Larger values increase cache hit rates and reduce mean response times, but increase problems with cache coherence. Note that sites that push (update) NIS maps nightly can set the value to be the equivalent of 12 hours or more with very good performance implications.

negative-time-to-live cachename value
Sets the time-to-live for negative entries (unsuccessful queries) in the specified cache. value is in integer seconds. Can result in significant performance improvements if there are several files owned by uids (user IDs) not in system databases; should be kept small to reduce cache coherency problems.

suggested-size cachename value
Sets the suggested number of hash buckets in the specified cache. This parameter should be changed only if the number of entries in the cache exceeds the suggested size by more than a factor of four or five. Since this is the internal hash table size, value should remain a prime number for optimum efficiency.

keep-hot-count cachename value
This attribute allows the administrator to set the number of entries nscd(1M) is to keep current in the specified cache. value is an integer number which should approximate the number of entries frequently used during the day.
check-files cachename value

Enables or disables checking the file belonging to the specified cachename for
changes. If enabled (which is the default), changes in the corresponding file cause
the cache to be invalidated within 10 seconds. Can be disabled if files are never
modified for a slight performance boost, particularly over NFS. value may be either
yes or no.

SEE ALSO
nscd(1M), group(4), hosts(4), ipnodes(4), passwd(4)

WARNINGS
The nscd.conf interface is included in this release on an uncommitted basis only
and is subject to change or removal in a future minor release.
nsswitch.conf – configuration file for the name service switch

/etc/nsswitch.conf

The operating system uses a number of databases of information about hosts, ipnodes, users (passwd and shadow), and groups. Data for these can come from a variety of sources: hostnames and host addresses, for example, can be found in /etc/hosts, NIS, NIS+, LDAP, or DNS. Zero or more sources may be used for each database; the sources and their lookup order are specified in the /etc/nsswitch.conf file.

The following databases use the switch file:

<table>
<thead>
<tr>
<th>Database</th>
<th>Used By</th>
</tr>
</thead>
<tbody>
<tr>
<td>aliases</td>
<td>sendmail(1M)</td>
</tr>
<tr>
<td>auth_attr</td>
<td>getauthnam(3SECDB)</td>
</tr>
<tr>
<td>automount</td>
<td>automount(1M)</td>
</tr>
<tr>
<td>bootparams</td>
<td>rpc.bootparamd(1M)</td>
</tr>
<tr>
<td>ethers</td>
<td>ethers(3SOCKET)</td>
</tr>
<tr>
<td>group</td>
<td>getgrnam(3C)</td>
</tr>
<tr>
<td>hosts</td>
<td>gethostbyname(3NSL). See Interaction with netconfig.</td>
</tr>
<tr>
<td>ipnodes</td>
<td>getipnodebyname(3SOCKET)</td>
</tr>
<tr>
<td>netgroup</td>
<td>innetgr(3C)</td>
</tr>
<tr>
<td>netmasks</td>
<td>ifconfig(1M)</td>
</tr>
<tr>
<td>networks</td>
<td>getnetbyname(3SOCKET)</td>
</tr>
<tr>
<td>passwd</td>
<td>getpwnam(3C), getspnam(3C), getauusernam(3BSM), getusername(3SECDB)</td>
</tr>
<tr>
<td>printers</td>
<td>lp(1), lpstat(1), cancel(1), lpr(1B), lpq(1B), lprm(1B), in.1pd(1M), lpadmin(1M), lpget(1M), lpset(1M)</td>
</tr>
<tr>
<td>prof_attr</td>
<td>getprofname(3SECDB), getexecprof(3SECDB)</td>
</tr>
<tr>
<td>project</td>
<td>getprojent(3EXACCT), getdefaultproj(3EXACCT), inproj(3EXACCT), newtask(1), setprojent(3EXACCT)</td>
</tr>
<tr>
<td>protocols</td>
<td>getprotobyname(3SOCKET)</td>
</tr>
<tr>
<td>publickey</td>
<td>getpublickey(3NSL), secure_rpc(3NSL)</td>
</tr>
<tr>
<td>rpc</td>
<td>getrpcbyname(3NSL)</td>
</tr>
</tbody>
</table>
Database Used By

sendmailvars sendmail(1M)
services getservbyname(3SOCKET).

See Interaction with netconfig.

The following sources may be used:

<table>
<thead>
<tr>
<th>Source</th>
<th>Uses</th>
</tr>
</thead>
<tbody>
<tr>
<td>files</td>
<td>/etc/hosts, /etc/passwd, /etc/inet/ipnodes, /etc/shadow</td>
</tr>
<tr>
<td>nis</td>
<td>NIS(YP)</td>
</tr>
<tr>
<td>nisplus</td>
<td>NIS+</td>
</tr>
<tr>
<td>ldap</td>
<td>LDAP</td>
</tr>
<tr>
<td>dns</td>
<td>Valid only for hosts; uses the Internet Domain Name Service.</td>
</tr>
<tr>
<td>compat</td>
<td>Valid only for passwd and group; implements &quot;+&quot; and &quot;-&quot;. See Interaction with +/− syntax.</td>
</tr>
<tr>
<td>user</td>
<td>Valid only for printers; implements support for ${HOME}/.printers.</td>
</tr>
<tr>
<td>xfn</td>
<td>Valid only for printers; implements support for FNS printer contexts. Provided to allow transition away from FNS printer contexts.</td>
</tr>
</tbody>
</table>

There is an entry in /etc/nsswitch.conf for each database. Typically these entries will be simple, such as "protocols: files" or "networks: files nisplus". However, when multiple sources are specified, it is sometimes necessary to define precisely the circumstances under which each source will be tried. A source can return one of the following codes:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUCCESS</td>
<td>Requested database entry was found.</td>
</tr>
<tr>
<td>UNAVAIL</td>
<td>Source is not configured on this system or internal failure.</td>
</tr>
<tr>
<td>NOTFOUND</td>
<td>Source responded &quot;no such entry&quot;</td>
</tr>
<tr>
<td>TRYAGAIN</td>
<td>Source is busy or not responding, might respond to retries.</td>
</tr>
</tbody>
</table>
For each status code, two actions are possible:

<table>
<thead>
<tr>
<th>Action</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>continue</td>
<td>Try the next source in the list.</td>
</tr>
<tr>
<td>return</td>
<td>Return now.</td>
</tr>
</tbody>
</table>

Additionally, for TRYAGAIN only, the following actions are possible:

<table>
<thead>
<tr>
<th>Action</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>forever</td>
<td>Retry the current source forever.</td>
</tr>
<tr>
<td>n</td>
<td>Retry the current source (n) more times, where (n) is an integer between 0 and MAX_INT (that is, 2.14 billion). After (n) retries has been exhausted, the action will continue to the next source.</td>
</tr>
</tbody>
</table>

The complete syntax of an entry is:

\[
<\text{entry}> ::= <\text{database}> "\!:\!" [<\text{source}> [\langle\text{criteria}\rangle]]*
\]

\[
<\text{criteria}> ::= "[" <\text{criterion}>+ "]"
\]

\[
<\text{criterion}> ::= <\text{status}>=<\text{action}>
\]

\[
<\text{status}>::= \text{success} | \text{notfound} | \text{unavail} | \text{tryagain}
\]

For every status except TRYAGAIN, the action syntax is:

\[
<\text{action}> ::= \text{return} | \text{continue}
\]

For the TRYAGAIN status, the action syntax is:

\[
<\text{action}> ::= \text{return} | \text{continue} | \text{forever} | \langle n \rangle
\]

\[
<\text{n}> ::= 0...\text{MAX\_INT}
\]

Each entry occupies a single line in the file. Lines that are blank, or that start with white space, are ignored. Everything on a line following a \# character is also ignored; the \# character can begin anywhere in a line, to be used to begin comments. The \<database> and \<source> names are case-sensitive, but \<action> and \<status> names are case-insensitive.

The library functions contain compiled-in default entries that are used if the appropriate entry in nsswitch.conf is absent or syntactically incorrect.

The default criteria for DNS and the NIS server in “DNS-forwarding mode” (and DNS server not responding or busy) is [SUCCESS=return NOTFOUND=continue UNAVAIL=continue TRYAGAIN=continue].
The default criteria for all other sources is [SUCCESS=return NOTFOUND=continue UNAVAIL=continue TRYAGAIN=forever].

The default, or explicitly specified, criteria are meaningless following the last source in an entry; and they are ignored, since the action is always to return to the caller irrespective of the status code the source returns.

In order to ensure that they all return consistent results, gethostbyname(3NSL), getipnodebyname(3SOCKET), getservbyname(3SOCKET), and netdir_getbyname(3NSL) functions are all implemented in terms of the same internal library function. This function obtains the system-wide source lookup policy for hosts, ipnodes, and services based on the inet family entries in netconfig(4) and uses the switch entries only if the netconfig entries have a "-" in the last column for nametoaddr libraries. See the NOTES section in gethostbyname(3NSL) and getservbyname(3SOCKET) for details.

The NIS+ server can be run in "YP-compatibility mode", where it handles NIS (YP) requests as well as NIS+ requests. In this case, the clients get much the same results (except for getspnam(3C)) from the "nis" source as from "nisplus"; however, "nisplus" is recommended instead of "nis".

The NIS (YP) server can be run in "DNS-forwarding mode", where it forwards lookup requests to DNS for host-names and -addresses that do not exist in its database. In this case, specifying "nis" as a source for "hosts" is sufficient to get DNS lookups; "dns" need not be specified explicitly as a source.

In SunOS 5.3 (Solaris 2.3) and compatible versions, the NIS+ server in "NIS/YP-compatibility mode" can also be run in "DNS-forwarding mode" (see rpc.nisd(1M)). Forwarding is effective only for requests originating from its YP clients; "hosts" policy on these clients should be configured appropriately.

When password aging is turned on, only a limited set of possible name services are permitted for the passwd: database in the /etc/nsswitch.conf file:

```
passwd: files
passwd: files nis
passwd: files nisplus
passwd: files ldap
passwd: compat
passwd_compat: nisplus
passwd_compat: ldap
```

Any other settings will cause the passwd(1) command to fail when it attempts to change the password after expiration and will prevent the user from logging in. These are the only permitted settings when password aging has been turned on. Otherwise,
you can work around incorrect passwd: lines by using the -r repository argument to the passwd(1) command and using passwd -r repository to override the nsswitch.conf settings and specify in which name service you want to modify your password.

Releases prior to SunOS 5.0 did not have the name service switch but did allow the user some policy control. In /etc/passwd one could have entries of the form +user (include the specified user from NIS passwd.byname), -user (exclude the specified user) and + (include everything, except excluded users, from NIS passwd.byname). The desired behavior was often "everything in the file followed by everything in NIS", expressed by a solitary + at the end of /etc/passwd. The switch provides an alternative for this case ("passwd: files nis") that does not require + entries in /etc/passwd and /etc/shadow (the latter is a new addition to SunOS 5.0, see shadow(4)).

If this is not sufficient, the NIS/YP compatibility source provides full +/- semantics. It reads /etc/passwd for getpwnam(3C) functions and /etc/shadow for getspnam(3C) functions and, if it finds +/- entries, invokes an appropriate source. By default, the source is "nis", but this may be overridden by specifying "nisplus" or "ldap" as the source for the pseudo-database passwd_compat.

Note that for every /etc/passwd entry, there should be a corresponding entry in the /etc/shadow file.

The NIS/YP compatibility source also provides full +/- semantics for group; the relevant pseudo-database is group_compat.

The compiled-in default entries for all databases use NIS (YP) as the enterprise level name service and are identical to those in the default configuration of this file:

```
passwd: files nis
group: files nis
hosts: nis [NOTFOUND=return] files
ipnodes: nis [NOTFOUND=return] files
networks: nis [NOTFOUND=return] files
protocols: nis [NOTFOUND=return] files
rpc: nis [NOTFOUND=return] files
ethers: nis [NOTFOUND=return] files
netmasks: nis [NOTFOUND=return] files
bootparams: nis [NOTFOUND=return] files
publickey: nis [NOTFOUND=return] files
netgroup: nis
```
The policy "nis [NOTFOUND=return] files" implies "if nis is UNAVAIL, continue on to files, and if nis returns NOTFOUND, return to the caller; in other words, treat nis as the authoritative source of information and try files only if nis is down." This, and other policies listed in the default configuration above, are identical to the hard-wired policies in SunOS releases prior to 5.0.

If compatibility with the +/- syntax for passwd and group is required, simply modify the entries for passwd and group to:

```
passwd: compat
group: compat
```

If NIS+ is the enterprise level name service, the default configuration should be modified to use nisplus instead of nis for every database on client machines. The file `/etc/nsswitch.nisplus contains a sample configuration that can be copied to `/etc/nsswitch.conf to set this policy.

If LDAP is the enterprise level name service, the default configuration should be modified to use ldap instead of nis for every database on client machines. The file `/etc/nsswitch.ldap contains a sample configuration that can be copied to `/etc/nsswitch.conf to set this policy.

If the use of +/- syntax is desired in conjunction with nisplus, use the following four entries:

```
passwd: compat
passwd_compat: nisplus OR ldap
group: compat
group_compat: nisplus OR ldap
```

In order to get information from the Internet Domain Name Service for hosts that are not listed in the enterprise level name service, NIS+ or LDAP, use the following configuration and set up the `/etc/resolv.conf file (see resolv.conf(4) for more details):
Many of the databases have enumeration functions: `passwd` has `getpwent()`, `hosts` has `gethostent()`, and so on. These were reasonable when the only source was `files` but often make little sense for hierarchically structured sources that contain large numbers of entries, much less for multiple sources. The interfaces are still provided and the implementations strive to provide reasonable results, but the data returned may be incomplete (enumeration for `hosts` is simply not supported by the `dns` source), inconsistent (if multiple sources are used), formatted in an unexpected fashion (for a host with a canonical name and three aliases, the `nisplus` source will return four hostents, and they may not be consecutive), or very expensive (enumerating a `passwd` database of 5,000 users is probably a bad idea). Furthermore, multiple threads in the same process using the same reentrant enumeration function (`getXXXent_r()` are supported beginning with SunOS 5.3) share the same enumeration position; if they interleave calls, they will enumerate disjoint subsets of the same database.

In general, the use of the enumeration functions is deprecated. In the case of `passwd`, `shadow`, and `group`, it may sometimes be appropriate to use `fgetgrent()`, `fgetpwent()`, and `fgetspent()` (see `getgrnam(3C)`, `getpwnam(3C)`, and `getspnam(3C)`, respectively), which use only the `files` source.

### FILES

A source named `SSS` is implemented by a shared object named `nss_SSS.so.1` that resides in `/usr/lib`.

- `/etc/nsswitch.conf`: Configuration file.
- `/usr/lib/nss_compat.so.1`: Implements "compat" source.
- `/usr/lib/nss_dns.so.1`: Implements "dns" source.
- `/usr/lib/nss_files.so.1`: Implements "files" source.
- `/usr/lib/nss_nis.so.1`: Implements "nis" source.
- `/usr/lib/nss_nisplus.so.1`: Implements "nisplus" source.
- `/usr/lib/nss_ldap.so.1`: Implements "ldap" source.
- `/usr/lib/nss_user.so.1`: Implements "user" source.
- `/usr/lib/nss_xfn.so.1`: Implements "xfn" source.
- `/etc/netconfig`: Configuration file for `netdir(3NSL)` functions that redirects hosts/devices policy to the switch.
- `/etc/nsswitch.files`: Sample configuration file that uses "files" only.
Sample configuration file that uses "files" and "nis".
Sample configuration file that uses "files" and "nisplus".
Sample configuration file that uses "files" and "ldap".
Sample configuration file that uses "files" and "dns" (but only for hosts).

SEE ALSO

ldap(1), newtask(1), nis+(1), passwd(1), automount(1M), ifconfig(1M),
rpc.bootparamd(1M), rpc.nisd(1M), sendmail(1M),
getauusername(3BSM) getgrent(3C), getnetgrent(3C), getpwnam(3C),
getspnam(3C), gethostbyname(3NSL), getpublickey(3NSL),
getrpcbyname(3NSL), netdir(3NSL), secure_rpc(3NSL),
getprojent(3EXACCT), getdefaultproj(3EXACCT), inproj(3EXACCT),
setprojent(3EXACCT), getauthnam(3SECDB), getexecprof(3SECDB),
getprofnam(3SECDB), getusernam(3SECDB), ethers(3SOCKET),
getipnodebyname(3SOCKET), getnetbyname(3SOCKET), netconfig(4),
project(4), resolv.conf(4), ypfiles(4)

NOTES

Within each process that uses nsswitch.conf, the entire file is read only once; if the
file is later changed, the process will continue using the old configuration.

Programs that use the getXXbyYY() functions cannot be linked statically since the
implementation of these functions requires dynamic linker functionality to access the
shared objects /usr/lib/nss_SSS.so.1 at run time.

The use of both nis and nisplus as sources for the same database is strongly
discouraged since both the name services are expected to store similar information
and the lookups on the database may yield different results depending on which name
service is operational at the time of the request. The same applies for using ldap
along with nis or nisplus.

Misspelled names of sources and databases will be treated as legitimate names of
(most likely nonexistent) sources and databases.

The following functions do not use the switch: fgetgrent(3C),
fgetprojent(3EXACCT), fgetpwnam(3C), fgetspent(3C), getpw(3C),
putpwent(3C), shadow(4).
### NAME
order – package installation order description file

### DESCRIPTION
The package installation order file, `.order`, is an ASCII file specifying the order in which packages must be installed based on their prerequisite dependencies. Any package with prerequisite dependencies must be installed _after_ any packages it lists as a prerequisite dependency in its `depend` file.

A `.order` file is required for the OS product. The `.order` file must reside in the top-level directory containing the product.

The ordering is specified as a list of package identifiers, from the first package to be installed to the last, one package identifier per line.

### NOTES
The `depend` file supports _incompatible_ and _reverse_ dependencies. These dependency types are not recognized in the `order` file.

### SEE ALSO
cdtoc(4), clustertoc(4), depend(4), packagetoc(4), pkginfo(4)

---

**NAME**

| order – package installation order description file |

**DESCRIPTION**
The package installation order file, `.order`, is an ASCII file specifying the order in which packages must be installed based on their prerequisite dependencies. Any package with prerequisite dependencies must be installed _after_ any packages it lists as a prerequisite dependency in its `depend` file.

A `.order` file is required for the OS product. The `.order` file must reside in the top-level directory containing the product.

The ordering is specified as a list of package identifiers, from the first package to be installed to the last, one package identifier per line.

**NOTES**
The `depend` file supports _incompatible_ and _reverse_ dependencies. These dependency types are not recognized in the `order` file.

**SEE ALSO**
cdtoc(4), clustertoc(4), depend(4), packagetoc(4), pkginfo(4)
ott(4)

<table>
<thead>
<tr>
<th>NAME</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ot−FACE object architecture information</td>
<td></td>
</tr>
</tbody>
</table>

| DESCRIPTION                                                                 |
| The FACE object architecture stores information about object-types in an ASCII file named .ott (object type table) that is contained in each directory. This file describes all of the objects in that directory. Each line of the .ott file contains information about one object in pipe-separated fields. The fields are (in order):
| name | the name of the actual system file. |
| dname | the name that should be displayed to the user, or a dot if it is the same as the name of the file. |
| description | the description of the object, or a dot if the description is the default (the same as object-type). |
| object-type | the FACE internal object type name. |
| flags | object specific flags. |
| mod time | the time that FACE last modified the object. The time is given as number of seconds since 1/1/1970, and is in hexadecimal notation. |
| object information | an optional field, contains a set of semi-colon separated name=value fields that can be used by FACE to store any other information necessary to describe this object. |

| FILES    | .ott is created in any directory opened by FACE. |

292  man pages section 4: File Formats • Last Revised 3 Jul 1990
The package table of contents file, `.packagetoc`, is an ASCII file containing all of the information necessary for installing a product release distributed in package form. It centralizes and summarizes all of the relevant information about each package in the product. This allows the install software to quickly read one file to obtain all of the relevant information about each package instead of having to examine each package at run time to obtain this information. The `.packagetoc` file resides in the top-level directory containing the product.

If a `.packagetoc` file exists for a product, there must also be a `.order` file.

Each entry in the `.packagetoc` file is a line that establishes the value of a parameter in the following form:

```PARAM=value```

A line starting with a pound-sign, “#”, is considered a comment and is ignored.

Parameters are grouped by package. The start of a package description is defined by a line of the form:

```PKG=value```

There is no order implied or assumed for specifying the parameters for a package with the exception of the `PKG` parameter, which must appear first. Only one occurrence of a parameter is permitted per package.

The parameters recognized are described below. Those marked with an asterisk are mandatory.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>PKG</code>*</td>
<td>The package identifier (for example, SUNWaccu). The maximum length of the identifier is nine characters. All the characters must be alphanumeric. The first character must be alphabetic. <code>install</code>, <code>new</code>, and <code>all</code> are reserved identifiers.</td>
</tr>
<tr>
<td><code>PKGDIR</code>*</td>
<td>The name of the directory containing the package. This directory is relative to the directory containing the product.</td>
</tr>
<tr>
<td><code>NAME</code>*</td>
<td>The full name of the package.</td>
</tr>
<tr>
<td><code>VENDOR</code></td>
<td>The name of the package’s vendor.</td>
</tr>
<tr>
<td><code>VERSION</code></td>
<td>The version of the package.</td>
</tr>
<tr>
<td><code>PRODNAME</code></td>
<td>The name of the product to which this package belongs.</td>
</tr>
</tbody>
</table>
**PRODVERS**

The version of the product to which this package belongs.

**SUNW_PKGTYPE**

The package type. Valid values are:

- **root**: indicates that the package will be installed in the `/` file system. The `root` packages are the only packages installed during dataless client installations. The `root` packages are spooled during a server installation to allow the later installation of diskless clients.

- **usr**: indicates that the package will be installed in the `/usr` file system.

- **kvm**: indicates that the package will be installed in the `/usr/platform` file system.

- **ow**: indicates a package that is part of the bundled OpenWindows product release. If no `SUNW_PKGTYPE` macro is present, the package is assumed to be of type `usr`.

**ARCH**

The architecture(s) supported by the package. This macro is taken from the package's `pkginfo(4)` file and is subject to the same length and formatting constraints.

The install program currently assumes that exactly one architecture token is specified for a package. For example, `ARCH=sparc.sun4c` is acceptable, but `ARCH=sparc.sun4c, sparc.sun4m` is not.

**DESC**

A detailed textual description of the package.

**BASEDIR**

The default installation base directory of the package.

**SUNW_PDEPEND**

A dependency specification for a prerequisite package. Each prerequisite dependency must appear as a separate macro. See `depend(4)` for more information on dependencies and instance specifications.

**SUNW_IDEPEnd**

A dependency specification for an incompatible package. Each incompatible dependency should appear as a separate macro. See `depend(4)` for more information on dependencies and instance specifications.

**SUNW_RDEPEnd**

A dependency specification for a reversed package dependency. Each reverse dependency should appear
as a separate macro. See `depend(4)` for more information on dependencies and instance specifications.

**CATEGORY**

The category of the package.

**SUNW_LOC**

Indicates that this package contains localizations for other packages. Such localization packages are treated as special case packages. Each package which has a `SUNW_LOC` macro must have a corresponding `SUNW_PKGLIST` macro. The value specified by this macro should be a valid locale.

**SUNW_PKGLIST**

A comma separated list of package identifiers. Currently this macro is used to indicate which packages are localized by a localization package.

**ROOTSIZE**

The space used by the package in the `/` file system.

**USRSIZE**

The space used by the package in the `/usr` subtree of the file system.

**VARSIZE**

The space used by the package in the `/var` subtree of the file system.

**OPTSIZE**

The space used by the package in the `/opt` subtree of the file system.

**EXPORTSIZE**

The space used by the package in the `/export` subtree of the file system.

**USROWSIZE**

The space used by the package in the `/usr/openwin` subtree of the file system.

**SPOOLEDSIZE**

The space used by the spooled version of this package. This is used during the setup of a server by the initial system installation programs.

All sizes are specified in bytes. Default disk partitions and file system sizes are derived from the values provided: accuracy is important.

**EXAMPLES**

**EXAMPLE 1** A sample `.packagetoc` file.

The following is an example package entry in a `.packagetoc` file.

```bash
#ident "@(#)packagetoc.4 1.2 92/04/28"
PKG=SUNWaccr
PKGDIR=SUNWaccr
NAME=System Accounting, (Root)
VENDOR=Sun Microsystems, Inc.
VERSION=8.1
PRODSNAME=SunOS
PRODSVERS=5.0beta2
SUNW_PKGTYPE=root
```

File Formats 295
EXAMPLE 1 A sample .packagetoc file. (Continued)

ARCH=sparc
DESC=System Accounting, (Root)
BASEDIR=/
CATEGORY=system
ROOTSIZE=11264
VARSIZE=15360
OPTSIZE=0
EXPORTSIZE=0
USRSIZE=0
USROWNSIZE=0

SEE ALSO

cdtoc(4), clustertoc(4), depend(4), order(4), pkginfo(4), pkgmap(4)

NOTES

The parameters NAME, VENDOR, VERSION, PRODNAME, PRODVERS, SUNW_PKGTYPE, SUNW_LOC, SUNW_PKGLIST, ARCH, DESC, BASEDIR, and CATEGORY are assumed to have been taken directly from the package's pkginfo(4) file. The length and formatting restrictions placed on the values for these parameters are identical to those for the corresponding entries in the pkginfo(4) file.

The value specified for the parameter PKGDIR should not exceed 255 characters.

The value specified for the parameters ROOTSIZE, VARSIZE, OPTSIZE, EXPORTSIZE, USRSIZE and USROWNSIZE must be a single integer value. The values can be derived from the package's pkgmap file by counting all space consumed by any files installed in the applicable file system. The space includes that used for directory entries and any UFS overhead that exists because of the way the files are represented (directory allocation scheme; direct, indirect, double indirect blocks; fragments; etc.)

The following kinds of entries in the pkgmap(4) file should be included in the space derivation:

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>regular file</td>
</tr>
<tr>
<td>c</td>
<td>character special file</td>
</tr>
<tr>
<td>b</td>
<td>block special file</td>
</tr>
<tr>
<td>p</td>
<td>pipe</td>
</tr>
<tr>
<td>l</td>
<td>hard link</td>
</tr>
<tr>
<td>s</td>
<td>symbolic link</td>
</tr>
<tr>
<td>x, d</td>
<td>directory</td>
</tr>
<tr>
<td>i</td>
<td>packaging installation script or information file (copyright, depend, postinstall, postremove)</td>
</tr>
</tbody>
</table>
packingrules – packing rules file for cachefs and filesync

DESCRIPTION

$HOME/.packingrules is a packing rules file for filesync and cachefspack. $HOME/.packingrules contains a list of directories and files that are to be packed and synchronized. It also contains a list of directories and files that are to be specifically excluded from packing and synchronization. See filesync(1) and cachefspack(1M).

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 manually create the packing rules file 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.

Blank lines and lines that begin with a pound sign (‘#’) are ignored.

Any line can be continued by placing a backslash (‘\’) immediately before the NEWLINE.

All other lines in the $HOME/.packingrules file have one of the following formats:

PACKINGRULES major.minor. This line is not actually required, but it should be the first line of every packing rules file. This line identifies the packing rules file for the file(1) command and specifies a format version number. The current version number is 1.1. See file(1).

BASE directory-1 [directory-2] This line identifies a directory (or pair of directories) under which files should be packed and synchronized. At least one directory name must be specified. For rules that are to be used by filesync a second directory name (where the copies are to be kept) must also be specified. The arguments must be fully qualified path names, and may include environment variables.

LIST name ... This line enumerates a list of files and sub-directories (beneath the current BASE) that are to be kept synchronized. This specification is recursive, in that specifying the name of a directory automatically
Includes all files and subdirectories it contains. Regular expressions (as described in glob and gmatch) are permitted. See glob(1) and gmatch(3GEN).

**IGNORE name ...**

This line enumerates a list of files that are not to be kept synchronized. Regular expressions (using glob and gmatch) are permitted.

There are important differences between the arguments to LIST and IGNORE statements. The arguments to a LIST statement can contain slashes and are interpreted as file names relative to the BASE directories. The arguments to an IGNORE statement are simpler names or expressions that cannot contain slashes. An IGNORE statement will not override a LIST statement. IGNORE statements only exclude files that are found beneath LISTed directories.

If the first name argument to a LIST statement begins with an exclamation point ('!'), the remainder of the statement will be executed as a command. The command will be run in the current BASE directory. The output of the command will be treated as a list of newline separated file names to be packed/synchronized. The resulting file names will be interpreted relative to the enclosing BASE directory.

If the first name argument to an IGNORE statement begins with an exclamation point ('!'), the remainder of the statement will be executed as a command. The command will be run in the current BASE directory. The command will be expected to figure out which names should not be synchronized. The output of the command will be treated as a list of newline separated file names that should be excluded from the packing and synchronization list.

Commands will be broken into distinct arguments and run directly with `sh -c`. Blanks can be embedded in an argument by escaping them with a backslash ('\') or enclosing the argument in double quotes ('"'). Double quotes can be passed in arguments by escaping the double quotes with a backslash ('\').

LIST lines only apply to the BASE statement that precedes them. IGNORE lines can appear before any BASE statement (in which case they apply to all BASEs) or after a BASE statement (in which case they only apply to the BASE that precedes them). Any number of these statements can occur in any combination. The order is not important.

**EXAMPLES**

**EXAMPLE 1** A sample $HOME.packingrules file.

The use of these statements is illustrated in the following $HOME.packingrules file.

```bash
# # junk files, not worth copying
# IGNORE core *.o *.bak *%
# # most of the stuff I want to keep in sync is in my $HOME
```
EXAMPLE 1  A sample $HOME.packingrules file.  (Continued)

BASE /net/bigserver/export/home/mynome $HOME
# everything in my work sub-directory should be maintained
LIST work
# a few of my favorite mail boxes should be replicated
LIST m/incoming
LIST m/action
LIST m/pending
#
# I like to carry around a couple of project directories
# but skip all the postscript output
#
BASE /net/bigserver/export/projects $HOME/projects
LIST poindexter epiphany
IGNORE *.ps
#
# the foonly package should always be kept on every machine
#
BASE /net/bigserver/opt/foonly /opt/foonly
LIST !cat .packinglist
#
# and the latest executables for the standard build environment
#
BASE /net/bigserver/export/buildenv $HOME/buildenv
LIST !find . -type f -a -perm -111 -a -print

SEE ALSO  file(1), filesync(1), cachefspack(1M)
pam.conf(4)

NAME  pam.conf – configuration file for pluggable authentication modules

SYNOPSIS /etc/pam.conf

DESCRIPTION  pam.conf is the configuration file for the Pluggable Authentication Module architecture, or PAM. A PAM module provides functionality for one or more of four possible services: authentication, account management, session management, and password management.

authentication service module         Provides functionality to authenticate a user and set up user credentials.
account management module       Provides functionality to determine if the current user’s account is valid. This includes checking for password and account expiration, as well as verifying access hour restrictions.
session management module       Provides functionality to set up and terminate login sessions.
password management module       Provides functionality to change a user’s authentication token or password.

Each of the four service modules can be implemented as a shared library object which can be referenced in the pam.conf configuration file.

The pam.conf file contains a listing of services. Each service is paired with a corresponding service module. When a service is requested, its associated module is invoked. Each entry has the following format:

service_name module_type control_flag module_path options

The following is an example of the pam.conf configuration file with support for authentication, account management, and session management modules.

login auth requisite 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
other session required pam_unix_session.so.1

service_name denotes the service (for example, login, dtlogin, or rlogin). The keyword, other, indicates the module all other applications which have not been specified should use.

The other keyword can also be used if all services of the same module_type have the same requirements.

In the example, since all of the services use the same session module, they could have been replaced by a single other line.
module_type denotes the service module type: authentication (auth), account management (account), session management (session), or password management (password).

The control_flag field determines the behavior of stacking.

The module_path field specifies the relative pathname to a shared library object which implements the service functionality. If the pathname is not absolute, it is assumed to be relative to /usr/lib/security/$ISA/.

The ISA token is replaced by an implementation defined directory name which defines the path relative to the calling program’s instruction set architecture.

The options field is used by the PAM framework layer to pass module specific options to the modules. It is up to the module to parse and interpret the options.

This field can be used by the modules to turn on debugging or to pass any module specific parameters such as a TIMEOUT value. It can also be used to support unified login. The options supported by the modules are documented in their respective manual pages.

When a service_name of the same module_type is defined more than once, the service is said to be stacked. Each module referenced in the module_path for that service is then processed in the order that it occurs in the configuration file. The control_flag field specifies the continuation and failure semantics of the modules, and can be requisite, required, optional, or sufficient.

The PAM framework processes each service module in the stack. If all requisite and required modules in the stack succeed, then success is returned, and optional and sufficient error values are ignored. If one or more requisite or required modules fail, then the error value from the first requisite or required module that failed is returned.

If none of the service modules in the stack are designated as requisite or required, then the PAM framework requires that at least one optional or sufficient module succeed.

If all fail then the error value from the first service module in the stack is returned.

The requisite and sufficient flags cause two exceptions to the above semantics. If a service module that is designated as requisite fails, then the PAM framework immediately returns an error to the application, and all subsequent service modules in the stack are ignored. If a prior required service module has failed, then that error is returned. If no prior required service module failed, then the error from the failed requisite service module is returned.

If a service module that is designated as sufficient succeeds, then the PAM framework immediately returns success to the application, and all subsequent service modules in the stack, even requisite and required ones, are ignored, given that all prior requisite and required modules have also succeeded. If a prior required module has failed, then the error value from that module is returned.
If any entry in *pam.conf* is incorrect, or if a module does not exist or cannot be opened, then all PAM services fail and users are not permitted access to the system. An error is logged through *syslog* (3C) at the LOG_CRIT level. To fix incorrect entries in *pam.conf*, a system administrator can boot the system in maintenance mode (single user) to edit the file.

The following is a sample configuration file that stacks the *su*, *login*, and *rlogin* services.

```plaintext
su  auth requisite    pam_inhouse.so.1
su  auth requisite    pam_authtok_get.so.1
su  auth required    pam_dhkeys.so.1
su  auth required    pam_unix_auth.so.1

login auth requisite    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 auth optional    pam_inhouse.so.1

rlogin auth sufficient    pam_rhosts_auth.so.1
rlogin auth requisite    pam_authtok_get.so.1
rlogin auth required    pam_dhkeys.so.1
rlogin auth required    pam_unix_auth.so.1
```

In the case of *su*, the user is authenticated by the inhouse and authtok_get, dhkeys, and unix_auth authentication modules. Because the inhouse and the other authentication modules are requisite and required, respectively, an error is returned back to the application if any module fails. In addition, if the requisite authentication (inhouse authentication) fails, the other authentication modules is never invoked, and the error is returned immediately back to the application.

In the case of *login*, the required keyword for control_flag requires that the user be allowed to login only if the user is authenticated by all the service modules. If authtok_get authentication fails, control continues to proceed down the stack, and the inhouse authentication module is invoked. inhouse authentication is optional by virtue of the optional keyword in the control_flag field. The user can still log in even if inhouse authentication fails, assuming the modules stacked above succeeded.

In the case of *rlogin*, the sufficient keyword for control_flag specifies that if the rhosts authentication check succeeds, then PAM should return success to rlogin and rlogin should not prompt the user for a password. The other authentication modules, which are in the stack, will only be invoked if the rhosts check fails. This gives the system administrator the flexibility to determine if rhosts alone is sufficient enough to authenticate a remote user.

Some modules return PAM_IGNORE in certain situations. In these cases the PAM framework ignores the entire entry in *pam.conf* regardless of whether or not it is requisite, required, optional or sufficient.
The following is a list of the utilities that use PAM: `login`, `passwd`, `su`, `rlogind`, `rshd`, `telnetd`, `ftpd`, `rpc.rexd`, `uucpd`, `init`, `sac`, `cron`, `ppp`, `dtsession`, `ssh` and `ttymon`.

The utility `dtlogin` also uses PAM. `dtlogin` is the login service utility for the Common Desktop Environment (CDE).

The PAM configuration file does not dictate either the name or the location of the service specific modules. The convention, however, is the following:

- `/usr/lib/security/$ISA/libpam.so.1` - File that implements the PAM framework library.
- `/etc/pam.conf` - Configuration file.

```
# PAM configuration
#
# Unless explicitly defined, all services use the modules
# defined in the "other" section.
#
# Modules are defined with relative pathnames, i.e., they are
# relative to /usr/lib/security/$ISA. Absolute path names, as
# present in this file in previous releases are still acceptable.
#
# Authentication management
#
# login service (explicit because of pam_dial_auth)
#
login auth requisite      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
#
# rlogin service (explicit because of pam_rhost_auth)
#
rlogin auth sufficient    pam_rhosts_auth.so.1
rlogin auth requisite     pam_authtok_get.so.1
rlogin auth required      pam_dhkeys.so.1
rlogin auth required      pam_unix_auth.so.1
#
# rsh service (explicit because of pam_rhost_auth)
# and pam_unix_auth for meaningful pam_setcred)
#
rsh  auth sufficient     pam_rhosts_auth.so.1
rsh  auth required       pam_authtok_get.so.1
```
EXAMPLE 1 A Sample pam.conf Configuration File  (Continued)

# # ppp service (explicit because of pam_dial_auth)
# ppp auth requisite pam_auth_tok_get.so.1
ppp auth required pam_dhkeys.so.1
ppp auth required pam_unix_auth.so.1
ppp auth required pam_dial_auth.so.1
#
# Default definitions for Authentication management
# Used when service name is not explicitly mentioned for authentication
# other auth requisite pam_auth_tok_get.so.1
other auth required pam_dhkeys.so.1
other auth required pam_unix_auth.so.1
#
# passwd command (explicit because of a different authentication module)
# passwd auth required pam_passwd_auth.so.1
#
# cron service (explicit because of non-usage of pam_roles.so.1)
# cron account required pam_projects.so.1
cron account required pam_unix_account.so.1
#
# Default definition for Account management
# Used when service name is not explicitly mentioned for account management
# other account requisite pam_roles.so.1
other account required pam_projects.so.1
other account required pam_unix_account.so.1
#
# Default definition for Session management
# Used when service name is not explicitly mentioned for session management
# other session required pam_unix_session.so.1
#
# Default definition for Password management
# Used when service name is not explicitly mentioned for password management
# other password required pam_dhkeys.so.1
other password requisite pam_auth_tok_get.so.1
other password requisite pam_auth_tok_check.so.1
other password required pam_auth_tok_store.so.1
#
# Support for Kerberos V5 authentication (uncomment to use Kerberos)
#
# rlogin auth optional pam_krb5.so.1 try_first_pass
# login auth optional pam_krb5.so.1 try_first_pass
# other auth optional pam_krb5.so.1 try_first_pass
# cron account optional pam_krb5.so.1
# other account optional pam_krb5.so.1
# other session optional pam_krb5.so.1
# other password optional pam_krb5.so.1 try_first_pass
ATTRIBUTES

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

<table>
<thead>
<tr>
<th>Attribute Type</th>
<th>Attribute Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MT Level</td>
<td>MT-Safe with exceptions</td>
</tr>
</tbody>
</table>

SEE ALSO

login(1), passwd(1), in.ftpd(1M), in.rlogind(1M), in.rshd(1M),
in.telnetd(1M), in.uucpd(1M), init(1M), rpc.rexd(1M), sac(1M),
ttymon(1M), su(1M), pam(3PAM), syslog(3C), libpam(3LIB), attributes(5),
environ(5), pam_authhtok_check(5), pam_authhtok_get(5),
pam_authhtok_store(5), pam_dhkeys(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_authhtok_check(5), pam_authhtok_get(5),
pam_authhtok_store(5), pam_dhkeys(5), pam_passwd_auth(5),
pam_unix_account(5), pam_unix_auth(5), and pam_unix_session(5).
passwd(4)

NAME passwd – password file

SYNOPSIS /etc/passwd

DESCRIPTION The file /etc/passwd is a local source of information about users’ accounts. The password file can be used in conjunction with other password sources, such as the NIS maps passwd.byname and passwd.bygid and the NIS+ table passwd. Programs use the getpwnam(3C) routines to access this information.

Each passwd entry is a single line of the form:

```
username:password:uid:gid:gos-field:home-dir:login-shell
```

where

username is the user’s login name. It is recommended that this field conform to the checks performed by pwck(1M).

password is an empty field. The encrypted password for the user is in the corresponding entry in the /etc/shadow file. pwconv(1M) relies on a special value of ‘x’ in the password field of /etc/passwd. If this value of ‘x’ exists in the password field of /etc/passwd, this indicates that the password for the user is already in /etc/shadow and should not be modified.

uid is the user’s unique numerical ID for the system.

gid is the unique numerical ID of the group that the user belongs to.

gcos-field is the user’s real name, along with information to pass along in a mail-message heading. (It is called the gcos-field for historical reasons.) An “&” (ampersand) in this field stands for the login name (in cases where the login name appears in a user’s real name).

home-dir is the pathname to the directory in which the user is initially positioned upon logging in.

login-shell is the user’s initial shell program. If this field is empty, the default shell is /usr/bin/sh.

The maximum value of the uid and gid fields is 2,147,483,647. To maximize interoperability and compatibility, administrators are recommended to assign users a range of UIDs and GIDs below 60,000 where possible.

The password file is an ASCII file. Because the encrypted passwords are always kept in the shadow file, /etc/passwd has general read permission on all systems and can be used by routines that map between numerical user IDs and user names.
Blank lines are treated as malformed entries in the passwd file and cause consumers of the file, such as getpwnam(3C), to fail.

Previous releases used a password entry beginning with a ‘+’ (plus sign) or ‘−’ (minus sign) to selectively incorporate entries from NIS maps for password. If still required, this is supported by specifying “passwd: compat” in nsswitch.conf(4). The “compat” source might not be supported in future releases. The preferred sources are files followed by the identifier of a name service, such as nis or ldap. This has the effect of incorporating the entire contents of the name service’s passwd database after the passwd file.

EXAMPLE 1 Sample passwd File

Here is a sample passwd file:

root:q.mJzTnu8icF.:0:10:God:/bin/csh
fred:6k/7KCFRNVXg:508:10: & Fredericks:/usr2/fred:/bin/csh

and the sample password entry from nsswitch.conf:

passwd: files nisplus

In this example, there are specific entries for users root and fred to assure that they can login even when the system is running single-user. In addition, anyone in the NIS+ table passwd will be able to login with their usual password, shell, and home directory.

If the password file is:

root:q.mJzTnu8icF.:0:10:God:/bin/csh
fred:6k/7KCFRNVXg:508:10: & Fredericks:/usr2/fred:/bin/csh
+

and the password entry from nsswitch.conf is:

passwd: compat

then all the entries listed in the NIS passwd.byuid and passwd.byname maps will be effectively incorporated after the entries for root and fred.
passwd(4)

FILES
/etc/nsswitch.conf
/etc/passwd
/etc/shadow

SEE ALSO
chgrp(1), chown(1), groups(1), login(1), newgrp(1), nispasswd(1), passwd(1),
sh(1), sort(1), chown(1M), domainname(1M), getent(1M), in.ftpd(1M),
passmsgmt(1M), pwck(1M), pwconv(1M), su(1M), useradd(1M), userdel(1M),
usermod(1M), a64l(3C), crypt(3C), getpw(3C), getpwnam(3C), getsnamp(3C),
putpwent(3C), group(4), hosts.equiv(4), nsswitch.conf(4), shadow(4),
environ(5), unistd(3HEAD)

System Administration Guide, Volume 1
NAME  pathalias – alias file for FACE

SYNOPSIS  /usr/vmsys/pathalias

DESCRIPTION  The pathalias files contain lines of the form alias=path where path can be one or more colon-separated directories. Whenever a FACE (Framed Access Command Environment, see face(1)) user references a path not beginning with a “/”, this file is checked. If the first component of the pathname matches the left-hand side of the equals sign, the right-hand side is searched much like $PATH variable in the system. This allows users to reference the folder $HOME/FILECABINET by typing filecabinet.

There is a system-wide pathalias file called $VMSYS/pathalias, and each user can also have local alias file called $HOME/pref/pathalias. Settings in the user alias file override settings in the system-wide file. The system-wide file is shipped with several standard FACE aliases, such as filecabinet, wastebasket, preferences, other_users, etc.

FILES  $HOME/pref/pathalias
       $VMSYS/pathalias

SEE ALSO  face(1)

NOTES  Unlike command keywords, partial matching of a path alias is not permitted, however, path aliases are case insensitive. The name of an alias should be alphabetic, and in no case can it contain special characters like “/”, “\”, or “=” . There is no particular limit on the number of aliases allowed. Alias files are read once, at login, and are held in core until logout. Thus, if an alias file is modified during a session, the change will not take effect until the next session.
path_to_inst(4)

NAME
path_to_inst – device instance number file

SYNOPSIS
/etc/path_to_inst

DESCRIPTION
/etc/path_to_inst records mappings of physical device names to instance numbers.

The instance number of a device is encoded in its minor number, and is the way that a
device driver determines which of the possible devices that it may drive is referred to
by a given special file.

In order to keep instance numbers persistent across reboots, the system records them
in /etc/path_to_inst.

This file is read only at boot time, and is updated by add_drv(1M) and
drvconfig(1M).

Note that it is generally not necessary for the system administrator to change this file,
as the system will maintain it.

The system administrator can change the assignment of instance numbers by editing
this file and doing a reconfiguration reboot. However, any changes made in this file
will be lost if add_drv(1M) or drvconfig(1M) is run before the system is rebooted.

Each instance entry is a single line of the form:
"physical name" instance number "driver binding name"

where

physical name is the absolute physical pathname of a device. This
pathname must be enclosed in double quotes.

instance number is a decimal or hexadecimal number.

driver binding name is the name used to determine the driver for the device.
This name may be a driver alias or a driver name. The
driver binding name must be enclosed in double
quotes.

EXAMPLES

EXAMPLE 1 Sample path_to_inst Entries

Here are some sample path_to_inst entries:

"/iommu@f,e00000000 0 "iommu"
"/iommu@f,e00000000/sbus@f,e00010000 0 "sbus"
"/iommu@f,e00000000/sbus@f,e00010000/sbusmem@e,0" 14 "sbusmem"
"/iommu@f,e00000000/sbus@f,e00010000/sbusmem@f,0" 15 "sbusmem"
"/iommu@f,e00000000/sbus@f,e00010000/ledma@f,0000010000 0 "ledma"
"/obio/serial@0,100000 0 "zs"
"/SUNW,sx@f,80000000" 0 "SUNW,sx"
EXAMPLE 1 Sample path_to_inst Entries  (Continued)

FILES
/etc/path_to_inst

SEE ALSO
add_drv(1M), boot(1M), drvconfig(1M), mknod(1M)

WARNINGS
If the file is removed the system may not be bootable (as it may rely on information found in this file to find the root, usr or swap device). If it does successfully boot, it will regenerate the file, but after rebooting devices may end up having different minor numbers than they did before, and special files created via mknod(1M) may refer to different devices than expected.

For the same reasons, changes should not be made to this file without careful consideration.

NOTES
This document does not constitute an API. path_to_inst may not exist or may have a different content or interpretation in a future release. The existence of this notice does not imply that any other documentation that lacks this notice constitutes an API.
The Peripheral Component Interconnect (PCI) bus is a little endian bus. PCI devices are self-identifying — that is to say the PCI device provides configuration parameters to the system which allows the system to identify the device and its driver. The configuration parameters are represented in the form of name-value pairs that can be retrieved using the DDI property interfaces. See `ddi_prop_lookup(9F)` for details.

The PCI bus properties are derived from PCI Configuration Space, or supplied by the Fcode PROM if it exists. Therefore, driver configuration files are not necessary for these devices.

However, on some occasions, drivers for PCI devices may use driver configuration files to provide driver private properties. This can be done through global property mechanism. See `driver.conf(4)` for further details. Driver configuration files can also be used to augment or override properties for a specific instance of a driver.

All bus drivers of class `pci` recognize the following properties:

**reg**

An arbitrary length array where each element of the array consists of a 5-tuple of 32-bit values. Each array element describes a logically contiguous mappable resource on the PCI bus.

The first 3 values in the 5-tuple describe the PCI address of the mappable resource. The first tuple contains the following information:

| Bits 0 - 7 | 8-bit Register number |
| Bits 8 - 10 | 3-bit Function number |
| Bits 11 - 15 | 5-bit Device number |
| Bits 16 - 23 | 8-bit Bus number |
| Bits 24 - 25 | 2-bit Address Space type identifier |

The Address Space type identifier may be interpreted as follows:

| 0x0 | Configuration Space |
| 0x1 | I/O Space |
| 0x2 | 32-bit Memory Space address |
| 0x3 | 64-bit Memory Space address |
The Bus number is a unique identifying number assigned to each PCI bus within a PCI domain.

The Device number is a unique identifying number assigned to each PCI device on a PCI bus. Note that a Device number is only unique within the set of Device numbers for a particular bus.

Each PCI device can have 1 to 8 logically independent functions, each with its own independent set of configuration registers. Each function on a device is assigned a Function number. For a PCI device with only one function, the Function number must be 0.

The Register number field selects a particular register within the set of configuration registers corresponding to the selected function.

The second and third values in the reg property 5-tuple specify the 64-bit address of the mappable resource within the PCI address domain. The second 32-bit tuple corresponds to the high order 4 bytes of the 64-bit address. The third 32-bit tuple corresponds to the low order bytes.

The fourth and fifth 32-bit values in the 5-tuple reg property specify the size of the mappable resource. The size is a 64-bit value where the fourth tuple corresponds to the high order bytes of the 64-bit size and the fifth corresponds to the low order.

The driver can refer to the elements of this array by index, and construct kernel mappings to these addresses using ddi_regs_map_setup(9F). The index into the array is passed as the rnumber argument of ddi_regs_map_setup(9F).

At a high-level interrupt context, you can use the ddi_get* and ddi_put* family of functions to access I/O and memory space. However, access to configuration space is not allowed when running at a high-interrupt level.

interrupts

This property consists of a single integer element array. Valid interrupt property values are 1, 2, 3, and 4. This value is derived directly from the contents of the device’s Configuration Interrupt Pin register.

A driver should use an index value of 0 when registering its interrupt handler with ddi_add_intr(9F).

All PCI devices support the reg property. The Device number and Function number as derived from the reg property are used to construct the address part of the device name under /devices.

Only devices that generate interrupts support an interrupts property.
Occasionally it may be necessary to override or augment the configuration information supplied by a PCI device. This can be achieved by writing a driver configuration file that describes a prototype device node specification containing the additional properties required.

For the system to merge the prototype node specification into an actual device node, certain conditions must be met. First, the name property must be identical. Second, the parent property must identify the PCI bus. Third, the unit-address property must identify the card. The format of the unit-address property is

\[ DD[,F] \]

where DD is the device number and F is the function number. If the function number is 0, only DD is specified.

EXAMPLE 1 A sample configuration file.

An example configuration file called `ACME,scsi-hba.conf` for a PCI driver called `ACME,scsi-hba` follows:

```
# Copyright (c) 1995, ACME SCSI Host Bus Adaptor
# ident "@(#)ACME,scsi-hba.conf 1.1 96/02/04"
name="ACME,scsi-hba" parent="/pci@1,0/pci@1f,4000"
    unit-address="3" scsi-initiator-id=6;
hba-advanced-mode="on";
hba-dma-speed=10;
```

In this example, we provide a property `scsi-initiator-id` to specify the SCSI bus initiator id that the adapter should use, for just one particular instance of adapter installed in the machine. We use the name property to identify the driver and the parent property to identify the particular bus the card is plugged into. This example uses the parent’s full path name to identify the bus. The unit-address property identifies the card itself, with device number of 3 and function number of 0.

Two global driver properties are also created: `hba-advanced-mode` (which has the string value `on`) and `hba-dma-speed` (which has the value 10 M bit/s). These properties apply to all device nodes of the `ACME,scsi-hba`. The following is an example configuration file called `ACME,foo.conf` for a PCI driver called `ACME,foo`:

```
# Copyright (c) 1996, ACME Foo driver
# ident "@(#)ACME,foo.conf 1.1 95/11/14"
name="ACME,foo" class="pci" unit-address="3,1"
    debug-mode=12;
```

In this example, we provide a property `debug-mode` for all instances of the `ACME,foo` driver with parents of class `pci` and device and function numbers of 3 and 1, respectively.
ATTRIBUTES

See 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>
</tbody>
</table>

SEE ALSO

driver.conf(4), attributes(5), ddi_add_intr(9F), ddi_prop_lookup(9F), ddi_rega_map_setup(9F)

Writing Device Drivers

IEEE 1275 PCI Bus Binding
pcmcia(4)

NAME  pcmcia – PCMCIA nexus driver

DESCRIPTION The PCMCIA nexus driver supports PCMCIA card client device drivers. There are no user-configurable options for this driver.

FILES /kernel/misc/pcmcia pcmcia driver

SEE ALSO pcmciad(1M)
The file /etc/phones contains the system-wide private phone numbers for the tip(1) program. /etc/phones is normally unreadable, and so may contain privileged information. The format of /etc/phones is a series of lines of the form:

<system-name> [ \t]*<phone-number>.

The system name is one of those defined in the remote(4) file and the phone number is constructed from [0123456789-*%]. The ‘*’ and ‘*’ characters are indicators to the auto call units to pause and wait for a second dial tone (when going through an exchange). The ‘*’ is required by the DF02-AC and the ‘*’ is required by the BIZCOMP 1030.

Comment lines are lines containing a ‘#’ sign in the first column of the line.

Only one phone number per line is permitted. However, if more than one line in the file contains the same system name tip(1) will attempt to dial each one in turn, until it establishes a connection.
pkginfo – package characteristics file

pkginfo is an ASCII file that describes the characteristics of the package along with information that helps control the flow of installation. It is created by the software package developer.

Each entry in the pkginfo file is a line that establishes the value of a parameter in the following form:

```
PARAM= "value"
```

There is no required order in which the parameters must be specified within the file. Each parameter is described below. Only fields marked with an asterisk are mandatory.

**PKG**

Abbreviation for the package being installed. All characters in the abbreviation must be alphanumeric and the first may not be numeric. The abbreviation is limited to a maximum length of nine characters. `install`, `new`, and `all` are reserved abbreviations. It is customary to make the first four letters unique to your company, such as the company's stock symbol.

**NAME**

Text that specifies the package name (maximum length of 256 ASCII characters). Use the `NAME` parameter as the foundation for describing the functionality and purpose of the package; spell out any acronyms and avoid internal product/project code names. The `DESC` parameter can then be used to expand the descriptive information. Use the `NAME` parameter to state as specifically as possible the use of the package, why a user would need to load it, and so on.

**ARCH**

A comma-separated list of alphanumeric tokens that indicate the architecture associated with the package. The `pkgmk(1)` tool may be used to create or modify this value when actually building the package. The maximum length of a token is 16 characters and it cannot include a comma.

Solaris 2 and Solaris 7's installation software meaningfully uses only one architecture token of the form:

```
<instruction_set_architecture> [. <platform_group>]
```

where `platform_group` is intended only for Solaris installation packages. Third party application software should restrict itself to `ARCH` values from the following Solaris-supported instruction set architectures (`uname -p`): `sparc`, `i386`, and `ppc`. Examples of Solaris' platform groups (`uname -m`) are `sun4u`, `sun4d`, and `sun4m` for the SPARC® instruction set and `i86pc` for the i386 instruction set. See `uname(1)` and `isalist(1)` for more details.
**VERSION**
Text that specifies the current version associated with the software package. The maximum length is 256 ASCII characters and the first character cannot be a left parenthesis. The `pkgmk(1)` tool may be used to create or modify this value when actually building the package. Current Solaris and Solaris-compatible software practice is to assign this parameter monotonically increasing Dewey decimal values of the form:

```
<major_revision>.<minor_revision> [.<micro_revision>]
```

where all the revision fields are integers. The versioning fields can be extended to an arbitrary string of numbers in Dewey-decimal format, if necessary.

**CATEGORY**
A comma-separated list of categories under which a package may be displayed. A package must at least belong to the system or application category. Categories are case-insensitive and may contain only alphanumerics. Each category is limited in length to 16 characters.

**DESC**
Text that describes the package (maximum length of 256 ASCII characters). This parameter value is used to provide the installer with a description of what the package contains and should build on the description provided in the NAME parameter. Try to make the two parameters work together so that a `pkginfo -1` will provide a fairly comprehensive textual description of the package.

**VENDOR**
Used to identify the vendor that holds the software copyright (maximum length of 256 ASCII characters).

**HOTLINE**
Phone number and/or mailing address where further information may be received or bugs may be reported (maximum length of 256 ASCII characters).

**EMAIL**
An electronic address where further information is available or bugs may be reported (maximum length of 256 ASCII characters).

**VSTOCK**
The vendor stock number, if any, that identifies this product (maximum length of 256 ASCII characters).

**CLASSES**
A space-separated list of classes defined for a package. The order of the list determines the order in which the classes are installed. Classes listed first will be installed first (on a media by media basis). This parameter may be modified by the request script.
**pkinfo(4)**

**ISTATES**
A list of allowable run states for package installation (for example, "S s 1" allows run states of S, s or 1). Solaris 2 and Solaris 7 support the run levels s, S, 0, 1, 2, 3, 5, and 6. Applicable run levels for this parameter are S, 1, 2, and 3. See init(1M) for details.

**RSTATES**
A list of allowable run states for package removal (for example, "S s 1" allows run states of S, s or 1). Solaris 2 and Solaris 7 support the run levels s, S, 0, 1, 2, 3, 5, and 6. Applicable run levels for this parameter are S, 1, 2, and 3. See init(1M) for details.

**BASEDIR**
The pathname to a default directory where "relocatable" files may be installed. If blank, the package is not relocatable and any files that have relative pathnames will not be installed. An administrator can override the default directory.

**ULIMIT**
If set, this parameter is passed as an argument to the ulimit(1) command (see limit(1)), which establishes the maximum size of a file during installation.

**ORDER**
A list of classes defining the order in which they should be put on the medium. Used by pkgmk(1) in creating the package. Classes not defined in this field are placed on the medium using the standard ordering procedures.

**MAXINST**
The maximum number of package instances that should be allowed on a machine at the same time. By default, only one instance of a package is allowed. This parameter must be set in order to have multiple instances of a package. In order to support multiple instances of packages (for example, packages that differ in their ARCH or VERSION parameter value), the value of this parameter must be high enough to allow for all instances of a given package, including multiple versions coexisting on a software server.

**PSTAMP**
Production stamp used to mark the pkgmap(4) file on the output volumes. Provides a means for distinguishing between production copies of a version if more than one is in use at a time. If PSTAMP is not defined, the default is used. The default consists of the UNIX system machine name followed by the string "YYYYMMDDHHMM" (year, month, date, hour, minutes).

**INTONLY**
Indicates that the package should only be installed interactively when set to any non-null value.

**SUNW_PRODNAME**
Solaris 2 and Solaris 7-only parameter indicating the name of the product this package is a part of or comprises (maximum length of 256 ASCII characters). A few examples of currently used SUNW_PRODNAME values are: "SunOS", "OpenWindows", and "Common Desktop Environment".
SUNW_PRODVERS
Solaris 2 and Solaris 7-only parameter indicating the version or release of the product described in SUNW_PRODNAME (maximum length of 256 ASCII characters). For example, where SUNW_PRODNAME="SunOS", and the Solaris 2.x Beta release, this string could be "5.x BETA", while for the Solaris 2.x FCS release, the string would be "5.x". For Solaris 7, the string is "5.7". If the SUNW_PRODNAME parameter is NULL, so should be the SUNW_PRODVERS parameter.

SUNW_PKGVERS
Solaris 2 and Solaris 7–only parameter indicating of version of the Solaris 2 or Solaris 7 package interface. It is used to indicate the version of the Solaris 2 or Solaris 7-specific software packaging interfaces.

SUNW_PKGVER="<sunw_package_version>"

where <sunw_package_version> has the form x.y[z] and x, y, and z are integers. For packages built for this release and previous releases, use SUNW_PKGVERS="1.0".

SUNW_PKGTYPE
Solaris 2 and Solaris 7–only parameter for Sun internal use only. Required for packages part of the Solaris 2 and Solaris 7 releases which install into the /, /usr, /usr/kvm, and /usr/openwin file systems. The Solaris 2 and Solaris 7 installation software must know which packages are part of which file system to properly install a server/client configuration. The currently allowable values for this parameter are root, usr, kvm, and ow. If no SUNW_PKGTYPE parameter is present, the package is assumed to be of BASEDIR= /opt. SUNW_PKGTYPE is optional only for packages which install into the /opt name space as is the case for the majority of Solaris 2 and Solaris 7-compatible add-on software. See the SUNW_PKGTYPE parameter in packagetoc(4) for further information.

SUNW_IsA
Solaris 2 and Solaris 7–only optional parameter that indicates a software package contains 64–bit objects if it is set to sparcv9. If this parameter is not set, the default ISA (instruction set architecture) is set to the value of the ARCH parameter.

SUNW_LOC
Solaris 2 and Solaris 7–only optional parameter used to indicate a software package containing localization files for a given product or application. The parameter value is a comma-separated list of locales supported by a package. It is only used for packages containing localization files, typically the message catalogues. The allowable values for this string field are those found in the table of Standard Locale Names located in the International Language Environments Guide.

SUNW_LOC="<locale_name>,<locale_name>,...,<locale_name>"

where

<locale_name>: = <language>[_[territory]] [.<codeset>]

<language>: = the set of names from ISO 639
<territory>::= the set of territories specified in ISO 3166
<codeset>::= is a string corresponding to the coded character set

Since a value of C specifies the traditional UNIX system behavior (American English, en_US), packages belonging to the C locale are viewed as non-localized packages, and thus must not have SUNW_LOC and SUNW_PKGLIST included in their pkginfo file. See also the SUNW_LOC parameter in packagetoc(4) and setlocale(3C) for more information. This keyword is not recognized by the add-on software utility Software Manager.

SUNW_PKGLIST
Solaris 2 and Solaris 7-only optional parameter used to associate a localization package to the package(s) from which it is derived. It is required whenever the SUNW_LOC parameter is defined. This parameter value is an comma-separated list of package abbreviations of the form:

SUNW_PKGLIST="pkg1[,version],pkg2[,version],..."

where version (if specified) should match the version string in the base package specified (see VERSION parameter in this manual page). When in use, SUNW_PKGLIST helps determine the order of package installation. The packages listed in the parameter will be installed before the localization package in question is installed. When left blank, SUNW_PKGLIST=" ", the package is assumed to be required for the locale to function correctly. See the SUNW_PKGLIST parameter in packagetoc(4) for more information. This keyword is not recognized by the add-on software utility Software Manager.

EXAMPLES

EXAMPLE 1 A Sample pkginfo File

The following is a sample pkginfo file:

SUNW_PRODNAME="SunOS"
SUNW_PRODVERS="5.5"
SUNW_PKGTYPE="usr"
PKG="SUNWesu"
NAME="Extended System Utilities"
VERSION="11.5.1"
ARCH="sparc"
VENDOR="Sun Microsystems, Inc."
HOTLINE="Please contact your local service provider"
EMAIL=""
VSTOCK="0122c3f5566"
CATEGORY="system"
ISTATES="S 2"
RSTATES="S 2"
ATTRIBUTES

See attributes(5) for 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>See entries below</td>
</tr>
<tr>
<td>PKG value</td>
<td>Evolving</td>
</tr>
<tr>
<td>VERSION value</td>
<td>Evolving</td>
</tr>
<tr>
<td>NAME value</td>
<td>Evolving</td>
</tr>
<tr>
<td>DESC value</td>
<td>Evolving</td>
</tr>
<tr>
<td>ARCH value</td>
<td>Evolving</td>
</tr>
<tr>
<td>CATEGORY value</td>
<td>Evolving</td>
</tr>
<tr>
<td>BASEDIR value</td>
<td>Evolving</td>
</tr>
<tr>
<td>ISTATES value</td>
<td>Evolving</td>
</tr>
<tr>
<td>RSTATES value</td>
<td>Evolving</td>
</tr>
<tr>
<td>MAXINST value</td>
<td>Evolving</td>
</tr>
<tr>
<td>SUNW_PRODNAME</td>
<td>Evolving</td>
</tr>
<tr>
<td>SUNW_PRODVERS</td>
<td>Evolving</td>
</tr>
<tr>
<td>SUNW_PKGVERS</td>
<td>Evolving</td>
</tr>
<tr>
<td>SUNW_PKGTYPE</td>
<td>Unstable</td>
</tr>
<tr>
<td>SUNW_LOC</td>
<td>Evolving</td>
</tr>
<tr>
<td>SUNW_PKGLIST</td>
<td>Evolving</td>
</tr>
</tbody>
</table>

SEE ALSO

isalist(1), limit(1), pkgmk(1), uname(1), init(1M), setlocale(3C), clustertoc(4), order(4), packagetoc(4), pkgmap(4), attributes(5)

Application Packaging Developer's Guide

International Language Environments Guide

NOTES

Developers may define their own installation parameters by adding a definition to this file. A developer-defined parameter must begin with a capital letter.

Trailing white space after any parameter value is ignored. For example, VENDOR="Sun Microsystems, Inc." is the same as VENDOR="Sun Microsystems, Inc.".
pkgmap(4)

NAME
pkgmap – package contents description file

DESCRIPTION
pkgmap is an ASCII file that provides a complete listing of the package contents. It is automatically generated by pkgmk(1) using the information in the prototype(4) file.

Each entry in pkgmap describes a single “deliverable object file.” A deliverable object file includes shell scripts, executable objects, data files, directories, and so forth. The entry consists of several fields of information, each field separated by a space. The fields are described below and must appear in the order shown.

part
An optional field designating the part number in which the object resides. A part is a collection of files and is the atomic unit by which a package is processed. A developer can choose the criteria for grouping files into a part (for example, based on class). If no value is defined in this field, part 1 is assumed.

ftype
A one-character field that indicates the file type. Valid values are:

b block special device
c character special device
d directory
e a file to be edited upon installation or removal (may be shared by several packages)
f a standard executable or data file
i installation script or information file
l linked file
p named pipe
s symbolic link
v volatile file (one whose contents are expected to change, like a log file)
x an exclusive directory accessible only by this package

class
The installation class to which the file belongs. This name must contain only alphanumeric characters and be no longer than 12 characters. It is not specified if the ftype is i (information file).

pathname
pathname may contain variables of the form $variable that support install-time configuration of the file. variable may be embedded in the pathname structure. (See prototype(4) for definitions of variable specifications.)

Do not use the following reserved words in pathname, since they are applied by pkgadd(1M) using a different mechanism:
### File Formats

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>major</td>
<td>The major device number. The field is only specified for block or character special devices.</td>
</tr>
<tr>
<td>minor</td>
<td>The minor device number. The field is only specified for block or character special devices.</td>
</tr>
<tr>
<td>mode</td>
<td>The octal mode of the file (for example, 0664). A question mark (?) indicates that the mode will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked files, packaging information files, or non-installable files. The mode can contain a variable specification. (See prototype(4) for definitions of variable specifications.)</td>
</tr>
<tr>
<td>owner</td>
<td>The owner of the file (for example, bin or root). The field is limited to 14 characters in length. A question mark (?) indicates that the owner will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked files or non-installable files. It is used optionally with a package information file. If used, it indicates with what owner an installation script will be executed. The owner can contain a variable specification. (See prototype(4) for definitions of variable specifications.)</td>
</tr>
<tr>
<td>group</td>
<td>The group to which the file belongs (for example, &quot;bin&quot; or &quot;sys&quot;). The field is limited to 14 characters in length. A question mark (?) indicates that the group will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked files or non-installable files. It is used optionally with a package information file. If used, it indicates with what group an installation script will be executed. The group can contain a variable specification. (See prototype(4) for definitions of variable specifications.)</td>
</tr>
<tr>
<td>size</td>
<td>The actual size of the file in bytes. This field is not specified for named pipes, special devices, directories or linked files.</td>
</tr>
<tr>
<td>cksum</td>
<td>The checksum of the file contents. This field is not specified for named pipes, special devices, directories, or linked files.</td>
</tr>
<tr>
<td>modtime</td>
<td>The time of last modification, as reported by the stat(2) function call. This field is not specified for named pipes, special devices, directories, or linked files.</td>
</tr>
</tbody>
</table>
Each pkgmap file must have one line that provides information about the number of parts, maximum size of parts that make up the package, and, optionally, the size of the package after compression (where size is given in 512-byte blocks). This line is in the following format:

```
: number_of_parts maximum_part_size compressed_pkg_size
```

Lines that begin with “#” are comment lines and are ignored.

When files are saved during installation before they are overwritten, they are normally just copied to a temporary pathname. However, for files whose mode includes execute permission (but which are not editable), the existing version is linked to a temporary pathname and the original file is removed. This allows processes which are executing during installation to be overwritten.

**EXAMPLE 1**

A sample pkgmap file

```
: 2 500
1 i pkginfo 237 1179 541296672
1 b class1 /dev/diskette 17 134 0644 root other
1 c class1 /dev/rdiskette 17 134 0644 root other
1 d none bin 0755 root bin
1 f none bin/INSTALL 0755 root bin 11103 17954 541295535
1 f none bin/REMOVE 0755 root bin 3214 50237 541295541
1 l none bin/UNINSTALL-bin/REMOVE
1 f none bin/cmda 0755 root bin 3580 60325 541295567
1 f none bin/cmdb 0755 root bin 49107 51255 541438368
1 f class1 bin/cmdc 0755 root bin 45599 26048 541295599
1 f class1 bin/cmdd 0755 root bin 4648 8473 541461238
1 f none bin/cmde 0755 root bin 40501 1264 541295622
1 f class2 bin/cmdf 0755 root bin 2345 35889 541295574
1 f none bin/cmdg 0755 root bin 41185 47653 541461242
2 d class2 data 0755 root bin
2 p class1 data/apipe 0755 root other
2 d none log 0755 root bin
2 v none log/logfile 0755 root bin 41815 47563 541461333
2 d none save 0755 root bin
2 d none spool 0755 root bin
2 d none tmp 0755 root bin
```

**SEE ALSO**

pkgmk(1), pkgadd(1M), stat(2), pkginfo(4), prototype(4)

**Application Packaging Developer’s Guide**

**NOTES**

The pkgmap file may contain only one entry per unique pathname.
The Solaris 2.5 release includes the .platform directory, a new directory on the Solaris CD image. This directory contains files (created by SunSoft and Solaris OEMs) that define platform support. These files are generically referred to as platform definition files. They provide a means to map different platform types into a platform group.

Platform definition files in the .platform directory are used by the installation software to ensure that software appropriate for the architecture of the system will be installed.

SunSoft provides a platform definition file named .platform/Solaris. This file is the only one that can define platform groups to which other platform definition files can refer. For example, an OEM platform definition file can refer to any platform group specified in the Solaris platform definition file.

Other platform definition files are delivered by OEMs. To avoid name conflicts, OEMs will name their platform definition file with an OEM-unique string. OEM’s should use whatever string they use to make their package names unique. This unique string is often the OEM’s stock symbol.

Comments are allowed in a platform definition file. A "#" begins a comment and can be placed anywhere on a line.

Platform definition files are composed of keyword-value pairs, and there are two kinds of stanzas in the file: platform group definitions and platform identifications.

- Platform group definitions:

The keywords in a platform group definition stanza are:

PLATFORM_GROUP The PLATFORM_GROUP keyword must be the first keyword in the platform group definition stanza. The value assigned to this keyword is the name of the platform group, for example:

PLATFORM_GROUP=sun4c

The PLATFORM_GROUP name is an arbitrary name assigned to a group of platforms. However, PLATFORM_GROUP typically equals the output of the uname -m command. PLATFORM_GROUP value cannot have white space and is limited to 256 ASCII characters.

INST_ARCH The instruction set architecture of all platforms in the platform group, for example:

INST_ARCH=sparc

The INST_ARCH keyword value must be the value returned by the uname -p command on all platforms in the platform group.
Platform identifications:

The keywords in a platform identification stanza are:

- **PLATFORM_NAME**
  - The `PLATFORM_NAME` keyword must be the first keyword in the platform identification stanza. The `PLATFORM_NAME` is the name assigned to the platform, for example:
    
    ```plaintext
    PLATFORM_NAME=SUNW,SPARCstation-5
    ```

    Typically, this name is the same as the value returned by the `uname -i` command on the machine, but it need not be the same.

    The `PLATFORM_NAME` value cannot have white space and is limited to 256 ASCII characters. If it contains parentheses, it must contain only balanced parentheses. For example, the string "foo(bar)foo" is a valid value for this keyword, but "foo(bar" is not.

    The other keywords in the platform identification stanza can be in any order, as long as the `PLATFORM_NAME` keyword is first.

- **PLATFORM_ID**
  - The value returned by the `uname -i` command on the machine, for example:
    
    ```plaintext
    PLATFORM_ID=SUNW,SPARCstation-5
    ```

- **MACHINE_TYPE**
  - The value returned by the `uname -m` command on the machine, for example:
    
    ```plaintext
    MACHINE_TYPE=sun4c
    ```

- **IN_PLATFORM_GROUP**
  - The platform group of which the platform is a member, for example:
    
    ```plaintext
    IN_PLATFORM_GROUP=sun4c
    ```

    The platform group name must be specified in the same file as the platform identification stanza or in the platform definition file with the name `platform/Solaris`.

    The `IN_PLATFORM_GROUP` keyword is optional. A platform doesn't have to belong to a platform group. If a platform isn't explicitly assigned to a platform group, it essentially forms its own platform group, where the platform group name is the `PLATFORM_NAME` value. The
IN_PLATFORM_GROUP value typically equals the output of the `uname -m` command. IN_PLATFORM_GROUP value cannot have white space and is limited to 256 ASCII characters.

INST_ARCH

The instruction set architecture of the platform, for example:

```
INST_ARCH=sparc
```

This field is only required if the platform does not belong to a platform group. The INST_ARCH keyword value must be the value returned by the `uname -p` command on all platforms in the platform group.

COMPATIBILITY

The installation program will remain compatible with the old Solaris CD format. If a Solaris CD image does not contain any platform definition files, the installation and upgrade programs will select the packages to be installed based on machine type (i.e., the value returned by the `uname -m` command).

EXAMPLES

EXAMPLE 1

The following example shows platform group definitions from the .platform/Solaris platform definition file.

```
# PLATFORM_GROUP=sun4c
INST_ARCH=sparc
#
PLATFORM_GROUP=sun4d
INST_ARCH=sparc
#
PLATFORM_GROUP=sun4m
INST_ARCH=sparc
#
PLATFORM_GROUP=sun4u
INST_ARCH=sparc
```

EXAMPLE 2

The following example shows platform identification stanzas, which define systems that belong in a platform group, from the .platform/Solaris platform definition file.

```
# PLATFORM_NAME=SUNW,Sun_4_20
PLATFORM_ID=SUNW,Sun_4_20
IN_PLATFORM_GROUP=sun4c
#
PLATFORM_NAME=SUNW,Sun_4_25
PLATFORM_ID=SUNW,Sun_4_25
IN_PLATFORM_GROUP=sun4c
#
PLATFORM_NAME=SUNW,SPARCstation-5
PLATFORM_ID=SUNW,SPARCstation-5
IN_PLATFORM_GROUP=sun4m
#
PLATFORM_NAME=SUNW,SPARCstation-10
```

File Formats 329
EXAMPLE 2 The following example shows platform identification stanzas, which define systems that belong in a platform group, from the .platform/Solaris platform definition file.  

PLATFORM_ID=SUNW,SPARCstation-10
IN_PLATFORM_GROUP=sun4m

FILES The .platform directory must reside as /cd_image/Solaris_vers/.platform, where

cd_image Is the path to the mounted Solaris CD (/cdrom/cdrom0/s0 by default) or the path to a copy of the Solaris CD on a disk.

Solaris_vers Is the version of Solaris: e.g., Solaris_2.5.

NOTES Typically, a platform identification stanza contains either a PLATFORM_ID or a MACHINE_TYPE stanza, but not both.

If both are specified, both must match for a platform to be identified as this platform type. Each platform identification stanza must contain either a PLATFORM_ID value or a MACHINE_TYPE value. If a platform matches two different platform identification stanzas—one which matched on the value of PLATFORM_ID and one which matched on the value of MACHINE_TYPE, the one that matched on PLATFORM_ID will take precedence.

The .platform directory is part of the Solaris CD image, whether that be the Solaris CD or a copy of the Solaris CD on a system’s hard disk.
plot – graphics interface

Files of this format are interpreted for various devices by commands described in plot(1B). A graphics file is a stream of plotting instructions. Each instruction consists of an ASCII letter usually followed by bytes of binary information. The instructions are executed in order. A point is designated by four bytes representing the x and y values; each value is a signed integer. The last designated point in an l, m, n, or p instruction becomes the "current point" for the next instruction.

m Move: the next four bytes give a new current point.

n Cont: draw a line from the current point to the point given by the next four bytes. See plot(1B).

p Point: plot the point given by the next four bytes.

l Line: draw a line from the point given by the next four bytes to the point given by the following four bytes.

t Label: place the following ASCII string so that its first character falls on the current point. The string is terminated by a NEWLINE.

a Arc: the first four bytes give the center, the next four give the starting point, and the last four give the end point of a circular arc. The least significant coordinate of the end point is used only to determine the quadrant. The arc is drawn counter-clockwise.

c Circle: the first four bytes give the center of the circle, the next two the radius.

e Erase: start another frame of output.

f Linemod: take the following string, up to a NEWLINE, as the style for drawing further lines. The styles are "dotted," "solid," "longdashed," "shortdashed," and "dotdashed." Effective only in plot 4014 and plot ver.

s Space: the next four bytes give the lower left corner of the plotting area; the following four give the upper right corner. The plot will be magnified or reduced to fit the device as closely as possible.

Space settings that exactly fill the plotting area with unity scaling appear below for devices supported by the filters of plot(1B). The upper limit is just outside the plotting area.

In every case the plotting area is taken to be square; points outside may be displayable on devices whose face is not square.

4014 space(0, 0, 3120, 3120);

ver space(0, 0, 2048, 2048);

300, 300s space(0, 0, 4096, 4096);
plot(4B)

SEE ALSO graph(1), plot(1B)
policy.conf – configuration file for security policy

/etc/security/policy.conf

The policy.conf file provides the security policy configuration for user-level attributes. Each entry consists of a key/value pair in the form:

key=value

The following key is defined:

AUTHS_GRANTED Specifies the default set of authorizations granted to all users. This entry is interpreted bychkauthattr(3SECDB). The value is one or more comma-separated authorizations defined inauth_attr(4).

The key/value pair must appear on a single line, and the key must start the line. Lines starting with # are taken as comments and ignored. Option name comparisons are case-insensitive.

EXAMPLE 1 Defining a key/value pair

AUTHS_GRANTED=com.sun.date

/ etc / user_attr Defines extended user attributes.

/ etc / security / auth_attr Defines authorizations.

/ etc / security / policy.conf Defines policy for the system.

SEE ALSO pfexec(1), chkauthattr(3SECDB), auth_attr(4), user_attr(4)
### NAME
power.conf – Power Management configuration information file

### SYNOPSIS
/etc/power.conf

### DESCRIPTION
The `power.conf` file is used by the Power Management configuration program `pmconfig(1M)` to initialize the settings for Power Management. If you make changes to this file, you must run `pmconfig(1M)` manually for the changes to take effect.

The `dtpower(1M)` GUI allows the configuration of a subset of parameters allowed by this file. For ease-of-use, it is recommended that you use `dtpower(1M)` to configure the parameters.

Power Management addresses two specific management scenarios: management of individual devices and management of the whole system. An individual device is power managed if a device supports multiple power levels and if the device driver uses Power Management interfaces provided by the kernel to save device power when the device is idle. If the driver uses the original Power Management interfaces, the device is controlled by the entries described in the DEVICE POWER MANAGEMENT section of this manual page. If the device driver uses new automatic device Power Management interfaces, the device is controlled by the entries described in the AUTOMATIC DEVICE POWER MANAGEMENT section of this manual page.

To determine if the device driver supports original Power Management interfaces, contact the device vendor. To find out if the device driver supports the new automatic device Power Management interfaces, look for “pm-components” property (`pm-components(9F)`) under the device name from the output of `prtconf -v` command (`prtconf(1M)`).

The original Power Management interfaces and the corresponding device Power Management entries in `power.conf` file that were supported in Solaris 7 and earlier releases are now obsolete. Support for them will be removed in a future release.

All entries in the `power.conf` file are processed in the order displayed in the file.

Device Power Management entries are now obsolete and support for them will be removed in a future release. If a device supports original Power Management interfaces, it needs to be explicitly configured for Power Management using an entry of the form shown below. A device will not be power managed if there is no entry for the device. Be sure you fully understand the Power Management framework before you attempt to modify device Power Management entries.

Device Power Management entries consist of line-by-line listings of the devices to be configured. Each line is of the form:

```
device_name threshold ...dependent_upon...
```
The fields must be in the order shown above. Each line must contain a device_name field and a threshold field; it may also contain a dependent_upon field. Fields and sub-fields are separated by white space (tabs or spaces). A line may be more than 80 characters. If a newline character is preceded by a backslash (\) it will be treated as white space. Comment lines must begin with a hash character (#).

The device_name field specifies the device to be configured. device_name is either a pathname specifying the device special file or a relative pathname containing the name of the device special file. (For the latter format, you can avoid using the full pathname by omitting the pathname component that specifies the parent devices. This includes the leading '/'.) Using the relative pathname format, the first device found with a full pathname containing device_name as its tail is matched. In either case, the leading /devices component of the pathname does not need to be specified.

The threshold field is used to configure the power manageable components of a device. These components represent entities within a device that may be power-managed separately. This field may contain as many integer values as the device has components. Each threshold time specifies the idle time in seconds before the respective component may be powered down. If there are fewer component threshold times than device components, the remaining components are not power managed. Use a value of −1 to explicitly disable power-down for a component. At least one component threshold must be specified per device (in the file).

The dependent_upon field contains a list of devices that must be idle and powered-down before the dependent device in device_name field can be powered down. A device must previously have been configured before it can be used in dependent_upon list. This field should only list logical dependents for this device. A logical dependent is a device that is not physically connected to the power managed device (for example, the display and the keyboard). Physical dependents are automatically considered and do not need to be included.

A device Power Management entry is only effective if there is no user process controlling the device directly. For example, X Window systems directly control framebuffers and entries in this file are effective only when X Windows are not running.

Devices whose drivers use the new automatic device Power Management interfaces (as evident by existence of pm-components(9) property) are automatically power managed if enabled by the autopm entry described below.

When a component has been idle at a given power level for its threshold time, the power level of the component will be reduced to the next lower power level of that component (if any). For devices which implement multiple components, each component is power-managed independently.

Default thresholds for components of automatically power managed devices are computed by the Power Management framework based on the system idleness threshold. By default, all components of the device are powered off if they have all been
idle for the system’s idleness threshold. The default system idleness threshold is determined by the applicable United States Environmental Protection Agency’s (EPA) Energy Star Memorandum of Understanding. See the NOTES section of this manual page for more information.

To set the system idleness threshold, use one of the following entries:

```
system-threshold threshold
system-threshold always-on
```

where *threshold* is the value of the system idleness threshold in hours, minutes or seconds as indicated by a trailing h, m or s (defaulting to seconds if only a number is given). If always-on is specified, then by default, all devices will be left at full power.

To override the default device component thresholds assigned by the Power Management framework, a device-thresholds entry may be used. A device-thresholds entry sets thresholds for a specific automatically power-managed device or disables automatic Power Management for the specific device.

A device-thresholds entry has the form:

```
device-thresholds phys_path (threshold ...) ...
```

or

```
device-thresholds phys_path threshold
```

or

```
device-thresholds phys_path always-on
```

where *phys_path* specifies the physical path (*libdevinfo*(3)) of a specific device. For example, `/pci@8,600000/scsi@4/ssd@w210000203700c3ee,0` specifies the physical path of a disk. A symbolic link into the /devices tree (for example `/dev/dsk/c1t1d0s0`) is also accepted. The thresholds apply (or keeping the device always on applies) to the specific device only.

In the first form above, each *threshold* value represents the number of hours, minutes or seconds (depending on a trailing h, m or s with a default to seconds) to spend idle at the corresponding power level before power will be reduced to the next lower level of that component. Parentheses are used to group thresholds per component, with the first (leftmost) group being applied to component 0, the next to component 1, etc. Within a group, the last (rightmost) number represents the time to be idle in the highest power level of the component before going to the next-to-highest level, while the first (leftmost) number represents the time to be idle in the next-to-lowest power level before going to the lowest power level.
If the number of groups does not match the number of components exported by the
device (via pm-components(9) property), or the number of thresholds in a group is
not one less than the number of power levels the corresponding component supports,
then an error message will be printed and the entry will be ignored.

For example, assume a device called xfb exports the components Frame Buffer and
Monitor. Component Frame Buffer has two power levels: Off and On. Component
Monitor has four power levels: Off, Suspend, Standby, and On.

The following device-thresholds entry:

device-thresholds /pci@f0000/xfb@0 (0) (3m 5m 15m)

would set the threshold time for the Monitor component of the specific xfb card to go
from On to Standby in 15 minutes, the threshold for Monitor to go from Standby to
Suspend in 5 minutes, and the threshold for Monitor to go from Suspend to Off in 3
minutes. The threshold for Frame Buffer to go from On to Off will be 0 seconds.

In the second form above, where a single threshold value is specified without
parentheses, the threshold value represents a maximum overall time within which the
entire device should be powered down if it is idle. Because the system does not know
about any internal dependencies there may be among a device's components, the
device may actually be powered down sooner than the specified threshold, but will not
take longer than the specified threshold, provided that all device components are idle.

In the third form above, all components of the device are left at full power.

Device Power Management entries are only effective if there is no user process
controlling the device directly. For example, X Window systems directly control frame
buffers and the entries in this file are effective only when X Windows are not running.

Dependencies among devices may also be defined. A device depends upon another if
none of its components may have their power levels reduced unless all components of
the other device are powered off. A dependency may be indicated by an entry of the
form:

device-dependency dependent_phys_path phys_path [ phys_path ... ]

where dependent_phys_path is the path name (as above) of the device that is kept up by
the others, and the phys_path entries specify the devices that keep it up. A symbolic
link into the /devices tree (such as /dev/fb) is also accepted. This entry is needed
only for logical dependents for the device. A logical dependent is a device that is not
physically connected to the power managed device (for example, the display and the
keyboard). Physical dependents are automatically considered and need not be
included.

An autopm entry may be used to enable or disable automatic device Power
Management on a system-wide basis. The format of the autopm entry is:

autopm behavior
### System Power Management

Acceptable behavior values and their meanings are:

<table>
<thead>
<tr>
<th>Behavior</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>The behavior of the system will depend upon its model. Desktop models that fall under the United States Environmental Protection Agency’s Energy Star Memorandum of Understanding #3 will have automatic device Power Management enabled, and all others will not. See the NOTES section of this manual page for more information.</td>
</tr>
<tr>
<td>enable</td>
<td>Automatic device Power Management will be started when this entry is encountered.</td>
</tr>
<tr>
<td>disable</td>
<td>Automatic device Power Management will be stopped when this entry is encountered.</td>
</tr>
</tbody>
</table>

The system Power Management entries control power management of the entire system using the suspend-resume feature. When the system is suspended, the complete current state is saved on the disk before power is removed. On reboot, the system automatically starts a resume operation and the system is restored to the state it was in prior to suspend.

The system can be configured to do an automatic shutdown (autoshutdown) using the suspend-resume feature by an entry of the following form:

`autoshutdown idle_time start_time finish_time behavior`

- **idle_time** specifies the time in minutes that the system must have been idle before it will be automatically shutdown. System idleness is determined by the inactivity of the system and can be configured as discussed below.

- **start_time** and **finish_time** (each in `hh:mm`) specify the time period during which the system may be automatically shutdown. These times are measured from the start of the day (12:00 a.m.). If the **finish_time** is less than or equal to the **start_time**, the period span from midnight to the **finish_time** and from the **start_time** to the following midnight. To specify continuous operation, the **finish_time** may be set equal to the **start_time**.

Acceptable **behavior** values and their meanings are:

<table>
<thead>
<tr>
<th>Behavior</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>shutdown</td>
<td>The system will be shut down automatically when it has been idle for the number of minutes specified in the <code>idle_time</code> value and the time of day falls between the <code>start_time</code> and <code>finish_time</code> values.</td>
</tr>
<tr>
<td>noshutdown</td>
<td>The system is never shut down automatically.</td>
</tr>
<tr>
<td>autowakeup</td>
<td>If the hardware has the capability to do autowakeup, the system is shut down as if the value were shutdown and the system will be restarted automatically the next time the time of day equals <code>finish_time</code>.</td>
</tr>
<tr>
<td>default</td>
<td>The behavior of the system will depend upon its model. Desktop models that fall under the United States Environmental Protection Agency’s Energy Star Memorandum of Understanding #3 will have automatic device Power Management enabled, and all others will not. See the NOTES section of this manual page for more information.</td>
</tr>
</tbody>
</table>
Agency's Energy Star Memorandum of Understanding #2 will have automatic shutdown enabled (as if behavior field were set to shutdown), and all others will not. See NOTES.

unconfigured The system will not be shut down automatically. If the system has just been installed or upgraded, the value of this field will be changed upon the next reboot.

You can use the following format to configure the system's notion of idleness:

\[ \text{idleness\_parameter} \text{ value} \]

Where idleness\_parameter can be:

**ttychars** If the idleness\_parameter is ttychars, the value field will be interpreted as the maximum number of tty characters that can pass through the \text{ldterm} module while still allowing the system to be considered idle. This value defaults to 0 if no entry is provided.

**loadaverage** If the idleness\_parameter is loadaverage, the (floating point) value field will be interpreted as the maximum load average that can be seen while still allowing the system to be considered idle. This value defaults to 0.04 if no entry is provided.

**diskreads** If the idleness\_parameter is diskreads, the value field will be interpreted as the maximum number of disk reads that can be perform by the system while still allowing the system to be considered idle. This value defaults to 0 if no entry is provided.

**nfsreqs** If the idleness\_parameter is nfsreqs, the value field will be interpreted as the maximum number of NFS requests that can be sent or received by the system while still allowing the system to be considered idle. Null requests, access requests, and getattr requests are excluded from this count. This value defaults to 0 if no entry is provided.

**idlecheck** If the idleness\_parameter is idlecheck, the value must be pathname of a program to be executed to determine if the system is idle. If autoshutdown is enabled and the console keyboard, mouse, tty, CPU (as indicated by load average), network (as measured by NFS requests) and disk (as measured by read activity) have been idle for the amount of time specified in the autoshutdown entry specified above, and the time of day falls between the start and finish times, then this program will be executed to check for other idleness criteria. The value

File Formats 339
of the idle time specified in the above autoshutdown entry will be passed to the program in the environment variable PM_IDLETIME. The process must terminate with an exit code that represents the number of minutes that the process considers the system to have been idle.

There is no default idlecheck entry.

When the system is suspended, the current system state is saved on the disk in a statefile. An entry of following form can be used to change the location of statefile:

statefile pathname

where pathname identifies a block special file, for example, /dev/dsk/c1t0d0s2, or is the absolute pathname of a local ufs file. If the pathname specifies a block special file, it can be a symbolic link as long as it does not have a file system mounted on it. If pathname specifies a local ufs file, it cannot be a symbolic link. If the file does not exist, it will be created during the suspend operation. All the directory components of the path must already exist.

The actual size of statefile depends on a variety of factors, including the size of system memory, the number of loadable drivers/modules in use, the number and type of processes running, and the amount of user memory that has been locked down. It is recommended that statefile be placed on a file system with at least 10 Mbytes of free space. In case there is no statefile entry at boot time, an appropriate new entry is automatically created by the 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>SUNWpmr</td>
</tr>
<tr>
<td>Interface stability</td>
<td>Evolving (Interfaces under DEVICE POWER MANAGEMENT are obsolete.)</td>
</tr>
</tbody>
</table>

SEE ALSO

pmconfig(1M), powerd(1M), sys-unconfig(1M), uadmin(2), attributes(5), cpr(7), ldterm(7M), pm(7D)

Writing Device Drivers

Using Power Management

NOTES

SPARC desktop models first shipped after October 1, 1995 and before July 1, 1999 comply with the United States Environmental Protection Agency’s Energy Star Memorandum of Understanding #2 guidelines and have autoshutdown enabled by default after 30 minutes of system idleness. This is achieved by default keyword of
autoshutdown entry behave as shutdown for these machines. The user is prompted to confirm this default behavior at system installation reboot, or during the first reboot after the system is unconfigured by sys-unconfig(1M).

SPARC desktop models first shipped after July 1, 1999 comply with the United States Environmental Protection Agency’s Energy Star Memorandum of Understanding #3 guidelines and have autoshutdown disabled by default, with autopm enabled after 30 minutes of idleness. This is achieved by interpreting default keyword of autopm entry behavior as enabled for these machines. User is not prompted to confirm this default behavior.

To determine the version of the EPA’s Energy Star Memorandum applicable to your machine, use:

```
prtconf -pv | grep -i energystar
```

Absence of a property indicates no Energy Star guidelines are applicable to your machine.

System Power Management (suspend-resume) is currently supported only on a limited set of hardware platforms. Please see the book Using Power Management for a complete list of platforms that support system Power Management. See uname(2) to programatically determine if the machine supports suspend-resume.
NAME
printers – user-configurable printer alias database

SYNOPSIS
$HOME/.printers

DESCRIPTION
The $HOME/.printers file is a simplified version of the system
/etc/printers.conf file (see printers.conf(4)). Users create the
$HOME/.printers file in their home directory. This optional file is customizable by
the user.

The $HOME/.printers file performs the following functions:

1. Sets personal aliases for all print commands.
2. Sets the interest list for the lpget, lpstat, and cancel commands. See
   lpget(1M), lpstat(1) and cancel(1).
3. Sets the default printer for the lp, lpr, lpq, and lprm commands. See lp(1),
   lpr(1B), lpq(1B), and lprm(1B).

Entries
Use a line or full screen editor to create or modify the $HOME/.printers file.

Each entry in $HOME/.printers describes one destination. Entries are one line
consisting of two fields separated by either BLANKs or TABs and terminated by a
NEWLINE. Format for an entry in $HOME/.printers varies according to the
purpose of the entry.

Empty lines can be included for readability. Entries may continue on to multiple lines
by adding a backslash (\) as the last character in the line. The $HOME/.printers
file can include comments. Comments have a pound sign (#) as the first character in
the line, and are terminated by a NEWLINE.

Setting Personal Aliases

Specify the alias or aliases in the first field. Separate multiple aliases by a pipe sign
(\'|\'). Specify the destination in the second field. A destination names a printer or class
of printers (see lpadmin(1M)). Specify the 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.

Setting the Interest List for lpget, lpstat and cancel

Specify _all in the first field. Specify the list of destinations for the interest list in the
second field. Separate each destinations by a comma (\',\'). Specify destinations using
atomic, POSIX-style (server: destination), or FNS names (.../service/printer/...).
See printers.conf(4) for information regarding the naming conventions for atomic
and FNS names. This list of destinations may refer to an alias defined in
$HOME/.printers.

Setting the Default Destination

342 man pages section 4: File Formats • Last Revised 10 Nov 1999
Locating Destination Information

Specify _default in the first field. Specify the default destination in the second field. Specify the default destination using atomic, POSIX-style (server:destination), or FNS names (.../service/printer/...). See printers.conf(4) for information regarding the naming conventions for atomic and FNS names. The default destination may refer to an alias defined in $HOME/.printers.

The print client commands locate destination information based on the “printers” database entry in the /etc/nsswitch.conf file. See nsswitch.conf(4).

Locating the Personal Default Destination

The default destination is located differently depending on the command.

The lp command locates the default destination in the following order:

1. lp command’s -d destination option.
2. LPDEST environment variable.
3. PRINTER environment variable.
4. _default destination in $HOME/.printers.
5. _default destination in /etc/printers.conf.
6. _default destination in FNS.

The lpr, lpq, and lprm commands locate the default destination in the following order:

1. lpr command’s -P destination option.
2. PRINTER environment variable.
3. LPDEST environment variable.
4. _default destination in $HOME/.printers.
5. _default destination in /etc/printers.conf.
6. _default destination in FNS.

Locating the Interest List for lpget, lpstat, and cancel

The lpget, lpstat, and cancel commands locate the interest list in the following order:

1. _all list in $HOME/.printers.
2. _all list in /etc/printers.conf.
3. _all list in FNS.

EXAMPLES

**EXAMPLE 1** Setting the interest list

The following entry sets the interest list to destinations ps, secure, and dog at server west and finance_ps at site bldg2:

```
_all ps,secure,west:dog,site/bldg2/service/printer/finance_ps
```

**EXAMPLE 2** Setting aliases to a printer

The following entry sets the aliases ps, lp, and lw to sparc_printer:

```
ps|lp|lw sparc_printer
```
EXAMPLE 2 Setting aliases to a printer

Continued

EXAMPLE 3 Setting an alias as a default destination

The following entry sets the alias pcl to hplj and sets it as the default destination:

```
pcl|_default hplj
```

EXAMPLE 4 Setting an alias to a server destination

The following entry sets the alias secure to destination catalpa at server tabloid:

```
secure tabloid:catalpa
```

EXAMPLE 5 Setting an alias to a site destination

The following entry sets the alias insecure to destination legal_ps at site bldg2:

```
insecure site/bldg2/service/printer/legal_ps
```

FILES

$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>
<tr>
<td>Stability Level</td>
<td>Stable</td>
</tr>
</tbody>
</table>

SEE ALSO

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

System Administration Guide, Volume 1

NOTES

$HOME/.printers is referenced by the printing commands before further name resolution is made in /etc/printers.conf or the name service. If the alias references a destination defined in /etc/printers.conf, it is possible that the destination is defined differently on different systems. This could cause output to be sent to an unintended destination if the user is logged in to a different system.
printers.conf – system printing configuration database

/etc/printers.conf

printers.conf.byname

printers.org_dir

fns.ctx_dir.domain

**NAME**

**SYNOPSIS**

**NIS**

**NIS+**

**FNS**

**DESCRIPTION**

The printers.conf file is the system printing configuration database. System administrators use printers.conf to describe destinations for the print client commands and the print protocol adapter. A destination names a printer or class of printers (see lpadmin(1M)). The LP print spooler uses private LP configuration data for represented in the printers.conf database.

**Entries**

Each entry in printers.conf describes one destination. Entries are one line consisting of any number of fields separated by colons (':') and terminated by a NEWLINE. The first field of each entry specifies the name of the destination and aliases to which the entry describes. Specify one or more names or aliases of the destination in this first field. Specify the destination using atomic names. POSIX-style names are not acceptable. See standards(5). Separate destination names by pipe signs ('|').

Two destination names are reserved for special use in the first entry. Use _all to specify the interest list for lpget, lpstat, and cancel. Use _default to specify the default destination.

The remaining fields in an entry are `key=value` pairs. See **Specifying Configuration Options** for details regarding `key=value` pairs.

Empty lines can be included for readability. Entries may continue on to multiple lines by adding a backslash ('\') as the last character in the line. printers.conf can include comments. Comments have a pound sign ('#') as the first character in the line, and are terminated by a NEWLINE. Use the lpset command to create or modify printers.conf (see lpset(1M)). Do not make changes in printers.conf using an editor.

`key=value` pairs are configuration options defined by the system administrator. `key` and `value` may be of arbitrary length. Separate `key` and `value` by the equal ('=' ) character.

**Client/Server Configuration Options**

The following client/server configuration options (represented as `key=value` pairs) are supported:

bsdaddr=server, destination[, Solaris]

Sets the server and destination name. Sets if the client generates protocol extensions for use with the `lp` command (see `lp(1)`). Solaris specifies a Solaris print server extension. If Solaris is not specified, no protocol extensions are generated. server is the name of the host containing the queue for destination. destination is the atomic name by which the server knows the destination.
use=destination
Sets the destination to continue searching for configuration information. destination is an atomic or Federated Naming Service (FNS) (.../service/printer/...) name.

all=destination_list
Sets the interest list for the lpget, lpstat, and cancel commands. destination_list is a comma-separated list of destinations. Specify destination using atomic or FNS names (.../service/printer/...). See lpget(1M), lpstat(1), and cancel(1).

General Server Options
The following general server configuration options (represented as key=value pairs) are supported:

spooling-type=spooler[.version]
Sets the type of spooler under which a destination is configured. Dynamically loads translation support for the back-end spooling system from
/usr/lib/print/bsd-adaptor/bsd_spooler.so[.version]. Specify spooler as lpsched, cascade, or test. lpsched is used as a default for locally attached destinations. cascade is used as a default for destination spooled on a remote host. Use test for the test module to allow the capture of print requests. If using a versioned spooler module, version specifies the version of the translation module.

spooling-type-path=dir_list
Sets the location of translation support for the type of spooler defined by the spooling-type key. Locates translation support for the for the type of spooler under which a destination is configured. dir_list is a comma-separated list of absolute pathnames to the directories used to locate translation support for the spooling system set by the spooling-type key.

LP Server Options
The following LP configuration options (represented as key=value pairs) are supported:

user-equivalence=true|false
Sets whether or not usernames are considered equivalent when cancelling a print request submitted from a different host in a networked environment. true means that usernames are considered equivalent, and permits users to cancel a print requests submitted from a different host. user-equivalence is set to false by default. false means that usernames are not considered equivalent, and does not permit users cancel a print request submitted from a different host. If user-equivalence is set to false, print requests can only be cancelled by the users on the host on whichs the print prequest was generated or by the super-user on the print server.

Test Configuration Options
The following test configuration options (represented as key=value pairs) are supported:
test-spooler-available=true|false
Sets whether or not the protocol adaptor accepts connection requests to the test adaptor for the destination. True means that the protocol adaptor accepts connection requests to the test adaptor for the destination. Test-spooler-available is set to true by default. False means that the protocol adaptor does not accept connection requests to the test adaptor for the destination.

test-log=dir
Sets the location of the log file generated by the test translation module. Specify dir as an absolute pathname.

test-dir=dir
Sets the directory to be used during execution of the test translation module. Specify dir as an absolute pathname.

test-access=true|false
Sets whether or not the requesting client has access to the test translation module. True means that the requesting client has access to the test translation module. Test-access is set to true by default. False means that the requesting client does not have access to the test translation module.

test-accepting=true|false
Sets whether or not the configured destination is accepting job submission requests. True means that the configured destination is accepting job submission requests. Test-accepting is set to true by default. False means that the configured destination is not accepting job submission requests.

test-restart=true|false
Sets whether or not a protocol request to restart the destination will be honored or return an error. True means that a protocol request to restart the destination will be honored. Test-restart is set to true by default. False means that a protocol request to restart the destination will return an error.

test-submit=true|false
Sets whether or not a protocol request to submit a job to a destination will be honored or return an error. True means that a protocol request to submit a job to a destination will be honored. Test-submit is set to true by default. False means that a protocol request to submit a job to a destination will not be honored.

test-show-queue-file=file
Sets the name of the file whose contents are to be returned as the result of a status query. Specify file as an absolute pathname.

test-cancel-cancel-file=file
Sets the name of the file whose contents are returned as the result of a cancellation request. Specify file as an absolute pathname.

The print client commands and the print protocol adaptor locate destination information based on the "printers" database entry in the /etc/nsswitch.conf file. See nsswitch.conf(4).
The default destination is located differently depending on the command.

The `lp` command locates the default destination in the following order:
1. `lp` command’s `-d destination` option.
2. LPDEST environment variable.
3. PRINTER environment variable.
4. `_default` destination in `$HOME/.printers`.
5. `_default` destination in `/etc/printers.conf`.
6. `_default` destination in FNS.

The `lpr`, `lpq`, and `lprm` commands locate the default destination in the following order:
1. `lpr` command’s `-P destination` option.
2. PRINTER environment variable.
3. LPDEST environment variable.
4. `_default` destination in `$HOME/.printers`.
5. `_default` destination in `/etc/printers.conf`.
6. `_default` destination in FNS.

Locating the Interest List for `lpstat`, `lpget`, and `cancel` commands locate the interest list in the following order:
1. _all list in `$HOME/.printers`.
2. _all list in `/etc/printers.conf`.
3. _all list in FNS.

Federated Naming Service (FNS) supports resolution of composite names spanning multiple naming systems. FNS supports several underlying naming services: NIS+, NIS, and files.

Atomic destination names are resolved using the search order specified by the “printers” database entry in the `/etc/nsswitch.conf` file. When the “xfn” service is configured in the “printers” database, the following Federated Name Service contexts are searched for the supplied name:

```
thisuser/service/printer,
myorgunit/service/printer,
```

**EXAMPLES**

**EXAMPLE 1** Setting the interest list

The following entry sets the interest list for the `lpget`, `lpstat` and `cancel` commands to `printer1`, `printer2` and `printer3`:

```
_all:all=printer1,printer2,printer3
```
EXAMPLE 1 Setting the interest list

EXAMPLE 2 Setting the server name

The following entry sets the server name to server and and printer name to ps_printer for destinations printer1 and ps. It does not generate protocol extensions.

printer1|ps:bsdaddr=server,ps_printer

EXAMPLE 3 Setting server name and destination name

The following entry sets the server name to server and destination name to pcl_printer, for destination printer2. It also generates Solaris protocol extensions.

printer2:bsdaddr=server,pcl_printer,Solaris

EXAMPLE 4 Setting server name and destination name with continuous search

The following entry sets the server name to server and destination name to new_printer, for destination printer3. It also sets the printer3 to continue searching for configuration information to printer another_printer.

printer3:bsdaddr=server,new_printer:use=another_printer

EXAMPLE 5 Setting default destination

The following entry sets the default destination to continue searching for configuration information to destination printer1.

_default:use=printer1

FILES

/etc/printers.conf
System configuration database.

$HOME/.printers
User-configurable printer database.

printers.conf.byname (NIS)
NIS version of /etc/printers.conf.

printers.org_dir (NIS+)
NIS+ version of /etc/printers.conf.

fns.ctx_dir.domain
FNS version of /etc/printers.conf.

/usr/lib/print/bsd-adaptor/bsd_spooler.so*
Spooler translation modules.

/usr/lib/print/in.lpd
BSD print protocol adapter.
ATTRIBUTES

See attributes(5) for 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>Stability Level</td>
<td>Stable</td>
</tr>
</tbody>
</table>

SEE ALSO
cancel(1), lp(1), lpq(1B), lpr(1B), lprm(1B), lpstat(1), in.lpd(1M), lpadm(1M), lpget(1M), lpset(1M), nsswitch.conf(4), printers(4), attributes(5), fns(5), fns_policies(5), standards(5)

System Administration Guide, Volume 1
/proc is a file system that provides access to the state of each process and light-weight process (lwp) in the system. The name of each entry in the /proc directory is a decimal number corresponding to a process-ID. These entries are themselves subdirectories. Access to process state is provided by additional files contained within each subdirectory; the hierarchy is described more completely below. In this document, “/proc file” refers to a non-directory file within the hierarchy rooted at /proc. The owner of each /proc file and subdirectory is determined by the user-ID of the process.

/proc can be mounted on any mount point, in addition to the standard /proc mount point, and can be mounted several places at once. Such additional mounts are allowed in order to facilitate the confinement of processes to subtrees of the file system via chroot(1M) and yet allow such processes access to commands like ps(1).

Standard system calls are used to access /proc files: open(2), close(2), read(2), and write(2) (including readv(2), writev(2), pread(2), and pwrite(2)). Most files describe process state and can only be opened for reading. ctl and lwptcl (control) files permit manipulation of process state and can only be opened for writing. as (address space) files contain the image of the running process and can be opened for both reading and writing. An open for writing allows process control; a read-only open allows inspection but not control. In this document, we refer to the process as open for reading or writing if any of its associated /proc files is open for reading or writing.

In general, more than one process can open the same /proc file at the same time. Exclusive open is an advisory mechanism provided to allow controlling processes to avoid collisions with each other. A process can obtain exclusive control of a target process, with respect to other cooperating processes, if it successfully opens any /proc file in the target process for writing (the as or ctl files, or the lwptcl file of any lwp) while specifying O_EXCL in the open(2). Such an open will fail if the target process is already open for writing (that is, if an as, ctl, or lwptcl file is already open for writing). There can be any number of concurrent read-only opens; O_EXCL is ignored on opens for reading. It is recommended that the first open for writing by a controlling process use the O_EXCL flag; multiple controlling processes usually result in chaos.

If a process opens one of its own /proc files for writing, the open succeeds regardless of O_EXCL and regardless of whether some other process has the process open for writing. Self-opens do not count when another process attempts an exclusive open. (A process cannot exclude a debugger by opening itself for writing and the application of a debugger cannot prevent a process from opening itself.) All self-opens for writing are forced to be close-on-exec (see the F_SETFD operation of fcntl(2)).

Data may be transferred from or to any locations in the address space of the traced process by applying lseek(2) to position the as file at the virtual address of interest followed by read(2) or write(2) (or by using pread(2) or pwrite(2) for the combined operation). The address-map file /proc/pid/map can be read to determine
the accessible areas (mappings) of the address space. I/O transfers may span contiguous mappings. An I/O request extending into an unmapped area is truncated at the boundary. A write request beginning at an unmapped virtual address fails with EIO; a read request beginning at an unmapped virtual address returns zero (an end-of-file indication).

Information and control operations are provided through additional files. `<procfs.h>` contains definitions of data structures and message formats used with these files. Some of these definitions involve the use of sets of flags. The set types `sigset_t`, `fltset_t`, and `sysset_t` correspond, respectively, to signal, fault, and system call enumerations defined in `<sys/signal.h>`, `<sys/fault.h>`, and `<sys/syscall.h>`. Each set type is large enough to hold flags for its own enumeration. Although they are of different sizes, they have a common structure and can be manipulated by these macros:

```c
prfillset(&set); /* turn on all flags in set */
premptyset(&set); /* turn off all flags in set */
praddset(&set, flag); /* turn on the specified flag */
prdelset(&set, flag); /* turn off the specified flag */
r = prismember(&set, flag); /* != 0 iff flag is turned on */
```

One of `prfillset()` or `premptyset()` must be used to initialize `set` before it is used in any other operation. `flag` must be a member of the enumeration corresponding to `set`.

Every process contains at least one light-weight process, or lwp. Each lwp represents a flow of execution that is independently scheduled by the operating system. All lwps in a process share its address space as well as many other attributes. Through the use of `lwpctl` and `ctl` files as described below, it is possible to affect individual lwps in a process or to affect all of them at once, depending on the operation.

When the process has more than one lwp, a representative lwp is chosen by the system for certain process status files and control operations. The representative lwp is a stopped lwp only if all of the process’s lwps are stopped; is stopped on an event of interest only if all of the lwps are so stopped (excluding PR_SUSPENDED lwps); is in a PR_REQUESTED stop only if there are no other events of interest to be found; or, failing everything else, is in a PR_SUSPENDED stop (implying that the process is deadlocked). See the description of the status file for definitions of stopped states. See the PCSTOP control operation for the definition of “event of interest”.

The representative lwp remains fixed (it will be chosen again on the next operation) as long as all of the lwps are stopped on events of interest or are in a PR_SUSPENDED stop and the PCRUN control operation is not applied to any of them.

When applied to the process control file, every /proc control operation that must act on an lwp uses the same algorithm to choose which lwp to act upon. Together with synchronous stopping (see PCSET), this enables a debugger to control a multiple-lwp process using only the process-level status and control files if it so chooses. More fine-grained control can be achieved using the lwp-specific files.
The system supports two process data models, the traditional 32-bit data model in which ints, longs and pointers are all 32 bits wide (the ILP32 data model), and on some platforms the 64-bit data model in which longs and pointers, but not ints, are 64 bits in width (the LP64 data model). In the LP64 data model some system data types, notably `size_t`, `off_t`, `time_t` and `dev_t`, grow from 32 bits to 64 bits as well.

The `/proc` interfaces described here are available to both 32-bit and 64-bit controlling processes. However, many operations attempted by a 32-bit controlling process on a 64-bit target process will fail with `EOVERFLOW` because the address space range of a 32-bit process cannot encompass a 64-bit process or because the data in some 64-bit system data type cannot be compressed to fit into the corresponding 32-bit type without loss of information. Operations that fail in this circumstance include reading and writing the address space, reading the address-map file, and setting the target process’s registers. There is no restriction on operations applied by a 64-bit process to either a 32-bit or a 64-bit target processes.

The format of the contents of any `/proc` file depends on the data model of the observer (the controlling process), not on the data model of the target process. A 64-bit debugger does not have to translate the information it reads from a `/proc` file for a 32-bit process from 32-bit format to 64-bit format. However, it usually has to be aware of the data model of the target process. The `pr_dmodel` field of the `status` files indicates the target process's data model.

To help deal with system data structures that are read from 32-bit processes, a 64-bit controlling program can be compiled with the C preprocessor symbol `_SYSCALL32` defined before system header files are included. This makes explicit 32-bit fixed-width data structures (like `cstruct stat32`) visible to the 64-bit program. See `types32(3HEAD)`.

At the top level, the directory `/proc` contains entries each of which names an existing process in the system. These entries are themselves directories. Except where otherwise noted, the files described below can be opened for reading only. In addition, if a process becomes a `zombie` (one that has exited but whose parent has not yet performed a `wait(2)` upon it), most of its associated `/proc` files disappear from the hierarchy; subsequent attempts to open them, or to read or write files opened before the process exited, will elicit the error `ENOENT`.

Although process state and consequently the contents of `/proc` files can change from instant to instant, a single `read(2)` of a `/proc` file is guaranteed to return a sane representation of state; that is, the read will be atomic with respect to the state of the process. No such guarantee applies to successive reads applied to a `/proc` file for a running process. In addition, atomicity is not guaranteed for I/O applied to the `as` (address-space) file for a running process or for a process whose address space contains memory shared by another running process.

A number of structure definitions are used to describe the files. These structures may grow by the addition of elements at the end in future releases of the system and it is not legitimate for a program to assume that they will not.
A given directory /proc/pid contains the following entries. A process can use the invisible alias /proc/self if it wishes to open one of its own /proc files (invisible in the sense that the name “self” does not appear in a directory listing of /proc obtained from ls(1), getdents(2), or readdir(3C)).

As Contains the address-space image of the process; it can be opened for both reading and writing. lseek(2) is used to position the file at the virtual address of interest and then the address space can be examined or changed through read(2) or write(2) (or by using pread(2) or pwrite(2) for the combined operation).

cntl A write-only file to which structured messages are written directing the system to change some aspect of the process’s state or control its behavior in some way. The seek offset is not relevant when writing to this file. Individual lwps also have associated lwpctl files in the lwp subdirectories. A control message may be written either to the process’s ctl file or to a specific lwpctl file with operation-specific effects. The effect of a control message is immediately reflected in the state of the process visible through appropriate status and information files. The types of control messages are described in detail later. See CONTROL MESSAGES.

status Contains state information about the process and the representative lwp. The file contains a pstatus structure which contains an embedded lwpstatus structure for the representative lwp, as follows:

define struct pstatus {
    int pr_flags;            /* flags (see below) */
    int pr_nlwp;             /* number of lwps in the process */
    pid_t pr_pid;            /* process id */
    pid_t pr_ppid;           /* parent process id */
    pid_t pr_pgid;           /* process group id */
    pid_t pr_sid;            /* session id */
    id_t pr_aslwpid;         /* lwp-id of the aslwp, if any */
    id_t pr_agentid;         /* lwp-id of the agent lwp, if any */
    sigset_t pr_sigpend;     /* set of process pending signals */
    uintptr_t pr_brkbase;    /* virtual address of the process heap */
    size_t pr_brksize;       /* size of the process heap, in bytes */
    uintptr_t pr_stkbase;    /* virtual address of the process stack */
    size_t pr_stkszsize;     /* size of the process stack, in bytes */
    timestruc_t pr_utime;    /* process user cpu time */
    timestruc_t pr_stime;    /* process system cpu time */
    timestruc_t pr_cputime;  /* sum of children's user times */
    timestruc_t pr_cutime;   /* sum of children's system times */
    sigset_t pr_sigttrace;   /* set of traced signals */
    fltset_t pr_flttrace;    /* set of traced faults */
    sysset_t pr_sysentry;   /* set of system calls traced on entry */
    sysset_t pr_sysexit;    /* set of system calls traced on exit */
    char pr_dmodel;          /* data model of the process */
    taskid_t pr_taskid;      /* task id */
    projid_t pr_projid;      /* project id */
    lwpstatus_t pr_lwp;      /* status of the representative lwp */
} pstatus_t;

pr_flags is a bit-mask holding the following process flags. For convenience, it also contains the lwp flags for the representative lwp, described later.
The process is a system process (see PROCSTOP).

The process is the parent of a vforked child (see PCWATCH).

The process has its inherit-on-fork mode set (see PCSET).

The process has its run-on-last-close mode set (see PCSET).

The process has its kill-on-last-close mode set (see PCSET).

The process has its asynchronous-stop mode set (see PCSET).

The process has its asynchronous-stop mode set (see PCSET).

The process has its ptrace-compatibility mode set (see PCSET).

pr_nlwp is the total number of lwps in the process.

pr_pid, pr_ppid, pr_pgid, and pr_sid are, respectively, the process ID, the ID of the process’s parent, the process’s process group ID, and the process’s session ID.

pr_aslwpid is the lwp-ID for the "asynchronous signal lwp" (aslwp). It is zero if there is no aslwp in the process. The aslwp is the lwp designated to redirect asynchronous signals to other lwps in a multi-threaded process. See signal(3HEAD) for a description of the aslwp.

pr_agentid is the lwp-ID for the /proc agent lwp (see the PCAGENT control operation). It is zero if there is no agent lwp in the process.

pr_sigpend identifies asynchronous signals pending for the process.

pr_brkbase is the virtual address of the process heap and pr_brksize is its size in bytes. The address formed by the sum of these values is the process break (see brk(2)). pr_stkbase and pr_stksize are, respectively, the virtual address of the process stack and its size in bytes. (Each lwp runs on a separate stack; the distinguishing characteristic of the process stack is that the operating system will grow it when necessary.)

pr_utime, pr_stime, pr_cutime, and pr_cstime are, respectively, the user CPU and system CPU time consumed by the process, and the cumulative user CPU and system CPU time consumed by the process’s children, in seconds and nanoseconds.

pr_sigtrace and pr_flttrace contain, respectively, the set of signals and the set of hardware faults that are being traced (see PCSTRACE and PCSFAULT).

pr_sysentry and pr_sysexit contain, respectively, the sets of system calls being traced on entry and exit (see PCSENTRY and PCSEXIT).

pr_dmodel indicates the data model of the process. Possible values are:

- proc(4)
process data model is ILP32.

PR_MODEL_LP64 process data model is LP64.

PR_MODEL_NATIVE process data model is native.

The constant PR_MODEL_NATIVE reflects the data model of the controlling process, that is, its value is PR_MODEL_ILP32 or PR_MODEL_LP64 according to whether the controlling process has been compiled as a 32-bit program or a 64-bit program, respectively.

pr_lwp contains the status information for the representative lwp:

```c
typedef struct lwpstatus {
    int pr_flags; /* flags (see below) */
    id_t pr_lwpid; /* specific lwp identifier */
    short pr_why; /* reason for lwp stop, if stopped */
    short pr_what; /* more detailed reason */
    short pr_cursig; /* current signal, if any */
    siginfo_t pr_info; /* info associated with signal or fault */
    sigset_t pr_lwppend; /* set of signals pending to the lwp */
    sigset_t pr_lwphold; /* set of signals blocked by the lwp */
    struct sigaction pr_action; /* signal action for current signal */
    stack_t pr_altstack; /* alternate signal stack info */
    uintptr_t pr_oldcontext; /* address of previous ucontext */
    int pr_errno; /* errno for failed syscall */
    long pr_sysarg[PRSYSARGS]; /* arguments to this syscall */
    long pr_rval1; /* primary syscall return value */
    long pr_rval2; /* second syscall return value, if any */
    char pr_clname[PRCLSZ]; /* scheduling class name */
    timestruc_t pr_tstamp; /* real-time time stamp of stop */
    ulong_t pr_instr; /* current instruction */
    prgregset_t pr_reg; /* general registers */
    prfpregset_t pr_fpreg; /* floating-point registers */
} lwpstatus_t;
```

pr_flags is a bit-mask holding the following lwp flags. For convenience, it also contains the process flags, described previously.

- **PR_STOPPED** lwp is stopped.
- **PR_ISTOP** lwp is stopped on an event of interest (see PCSTOP).
- **PR_DSTOP** lwp has a stop directive in effect (see PCSTOP).
- **PR_STEP** lwp has a single-step directive in effect (see PCRUN).
- **PR_ASLZEPP** lwp is in an interruptible sleep within a system call.
- **PR_PGINVAL** lwp’s current instruction (pr_instr) is undefined.
- **PR_ASLWP** this is the asynchronous signal lwp for the process.
- **PR_AGENT** this is the /proc agent lwp for the process.
pr_lwpid names the specific lwp.

pr_why and pr_what together describe, for a stopped lwp, the reason for the stop. Possible values of pr_why and the associated pr_what are:

- **PR_REQUESTED** indicates that the stop occurred in response to a stop directive, normally because PCSTOP was applied or because another lwp stopped on an event of interest and the asynchronous-stop flag (see PCSET) was not set for the process. pr_what is unused in this case.

- **PR_SIGNALLED** indicates that the lwp stopped on receipt of a signal (see PCSTRACE); pr_what holds the signal number that caused the stop (for a newly-stopped lwp, the same value is in pr_cursig).

- **PR_FAULTED** indicates that the lwp stopped on incurring a hardware fault (see PCSFAULT); pr_what holds the fault number that caused the stop.

- **PR_SYSENTRY** and **PR_SYSEXIT** indicate a stop on entry to or exit from a system call (see PCSENTRY and PCSEXIT); pr_what holds the system call number.

- **PR_JOBCONTROL** indicates that the lwp stopped due to the default action of a job control stop signal (see sigaction(2)); pr_what holds the stopping signal number.

- **PR_SUSPENDED** indicates that the lwp stopped due to internal synchronization of lwps within the process. pr_what is unused in this case.

pr_cursig names the current signal, that is, the next signal to be delivered to the lwp, if any. pr_info, when the lwp is in a PR_SIGNALLED or PR_FAULTED stop, contains additional information pertinent to the particular signal or fault (see <sys/siginfo.h>.

pr_lwp pend identifies any synchronous or directed signals pending for the lwp. pr_lwphold identifies those signals whose delivery is being blocked by the lwp (the signal mask).

pr_action contains the signal action information pertaining to the current signal (see sigaction(2)); it is undefined if pr_cursig is zero. pr_altstack contains the alternate signal stack information for the lwp (see sigaltstack(2)).

pr_oldcontext, if not zero, contains the address on the lwp stack of a ucontext structure describing the previous user-level context (see ucontext(3HEAD)). It is non-zero only if the lwp is executing in the context of a signal handler.
pr_syscall is the number of the system call, if any, being executed by the lwp; it is non-zero if and only if the lwp is stopped on PR_SYSENTRY or PR_SYSEXIT, or is asleep within a system call (PR_ASLEEP is set). If pr_syscall is non-zero, pr_nsysarg is the number of arguments to the system call and pr_sysarg contains the actual arguments.

pr_rval1, pr_rval2, and pr_errno are defined only if the lwp is stopped on PR_SYSEXIT or if the PR_VFORKP flag is set. If pr_errno is zero, pr_rval1 and pr_rval2 contain the return values from the system call. Otherwise, pr_errno contains the error number for the failing system call (see <sys/errno.h>).

pr_clname contains the name of the lwp’s scheduling class.

pr_tstamp, if the lwp is stopped, contains a time stamp marking when the lwp stopped, in real time seconds and nanoseconds since an arbitrary time in the past.

pr_instr contains the machine instruction to which the lwp’s program counter refers. The amount of data retrieved from the process is machine-dependent. On SPARC based machines, it is a 32-bit word. On IA based machines, it is a single byte. In general, the size is that of the machine’s smallest instruction. If PR_PCINVAL is set, pr_instr is undefined; this occurs whenever the lwp is not stopped or when the program counter refers to an invalid virtual address.

pr_reg is an array holding the contents of a stopped lwp’s general registers.

SPARC

On SPARC-based machines, the predefined constants R_G0 ... R_G7, R_O0 ... R_O7, R_L0 ... R_L7, R_I0 ... R_I7, R_PC, R_nPC, and R_Y can be used as indices to refer to the corresponding registers; previous register windows can be read from their overflow locations on the stack (however, see the gwindows file in the /proc/pid/lwp/lwpid subdirectory).

SPARC V8 (32-bit)

For SPARC V8 (32-bit) controlling processes, the predefined constants R_PSR, R_WIM, and R_TBR can be used as indices to refer to the corresponding special registers. For SPARC V9 (64-bit) controlling processes, the predefined constants R_CCR, R_ASI, and R_FPRS can be used as indices to refer to the corresponding special registers.

IA

On IA based machines, the predefined constants SS, UESP, EFL, CS, EIP, ERR, TRAPNO, EAX, ECX, EDX, EBX, ESP, EBP, ESI, EDI, DS, ES, FS, and GS can be used as indices to refer to the corresponding registers.

pr_fpreg is a structure holding the contents of the floating-point registers.
SPARC registers, both general and floating-point, as seen by a 64-bit controlling process are the V9 versions of the registers, even if the target process is a 32-bit (V8) process. V8 registers are a subset of the V9 registers.

If the lwp is not stopped, all register values are undefined.

Contains miscellaneous information about the process and the representative lwp needed by the ps(1) command. psinfo is accessible after a process becomes a zombie. The file contains a psinfo structure which contains an embedded lwpsinfo structure for the representative lwp, as follows:

```c
typedef struct psinfo {
    int pr_flag; /* process flags */
    int pr_nlwp; /* number of lwps in the process */
    pid_t pr_pid; /* process id */
    pid_t pr_ppid; /* process id of parent */
    pid_t pr_pgid; /* process id of process group leader */
    pid_t pr_sid; /* session id */
    uid_t pr_uid; /* real user id */
    uid_t pr_euid; /* effective user id */
    gid_t pr_gid; /* real group id */
    gid_t pr_egid; /* effective group id */
    uintptr_t pr_addr; /* address of process */
    size_t pr_size; /* size of process image in Kbytes */
    size_t pr_rssize; /* resident set size in Kbytes */
    dev_t pr_ttydev; /* controlling tty device (or PRNODEV) */
    ushort_t pr_pctcpu; /* % of recent cpu time used by all lwps */
    ushort_t pr_pctmem; /* % of system memory used by process */
    timestruc_t pr_start; /* process start time, from the epoch */
    timestruc_t pr_time; /* cpu time for this process */
    timestruc_t pr_ctime; /* cpu time for reaped children */
    taskid_t pr_taskid; /* task id */
    projid_t pr_projid; /* project id */
    char pr_fname[PRFNSZ]; /* name of executed file */
    char pr_psargs[PRARGSZ]; /* initial characters of arg list */
    int pr_wstat; /* if zombie, the wait() status */
    int pr_argc; /* initial argument count */
    uintptr_t pr_argv; /* address of initial argument vector */
    uintptr_t pr_envp; /* address of initial environment vector */
    char pr_dmodel; /* data model of the process */
    lwpsinfo_t pr_lwp; /* information for representative lwp */
} psinfo_t;
```

Some of the entries in psinfo, such as pr_flag and pr_addr, refer to internal kernel data structures and should not be expected to retain their meanings across different versions of the operating system.

pr_pctcpu and pr_pctmem are 16-bit binary fractions in the range 0.0 to 1.0 with the binary point to the right of the high-order bit (1.0 == 0x8000). pr_pctcpu is the summation over all lwps in the process.

pr_lwp contains the ps(1) information for the representative lwp. If the process is a zombie, pr_nlwp and pr_lwp.pr_lwpid are zero and the other fields of pr_lwp are undefined.
typedef struct lwpsinfo {
    int pr_flag; /* lwp flags */
    id_t pr_lwpid; /* lwp id */
    uintptr_t pr_addr; /* internal address of lwp */
    uintptr_t pr_wchan; /* wait addr for sleeping lwp */
    char pr_stype; /* synchronization event type */
    char pr_state; /* numeric lwp state */
    char pr_sname; /* printable character for pr_state */
    char pr_nice; /* nice for cpu usage */
    short pr_syscall; /* system call number (if in syscall) */
    int pr_pri; /* priority, high value = high priority */
    ushort_t pr_pctcpu; /* % of recent cpu time used by this lwp */
    timestruc_t pr_start; /* lwp start time, from the epoch */
    char pr_clname[PRCLSZ]; /* scheduling class name */
    char pr_name[PRFNSZ]; /* name of system lwp */
    processorid_t pr_onpro; /* processor which last ran this lwp */
    processorid_t pr_bindpro; /* processor to which lwp is bound */
    psetid_t pr_bindpset; /* processor set to which lwp is bound */
} lwpsinfo_t;

Some of the entries in lwpsinfo, such as pr_flag, pr_addr, pr_wchan, pr_stype, pr_state, and pr_name, refer to internal kernel data structures and should not be expected to retain their meanings across different versions of the operating system.

pr_pctcpu is a 16-bit binary fraction, as described above. It represents the CPU time used by the specified lwp. On a multi-processor machine, the maximum value is 1/N, where N is the number of CPUs.

cred
Contains a description of the credentials associated with the process:

typedef struct prcred {
    uid_t pr_euid; /* effective user id */
    uid_t pr_ruid; /* real user id */
    uid_t pr_suid; /* saved user id (from exec) */
    gid_t pr_egid; /* effective group id */
    gid_t pr_rgid; /* real group id */
    gid_t pr_sgid; /* saved group id (from exec) */
    int pr_ngroups; /* number of supplementary groups */
    gid_t pr_groups[1]; /* array of supplementary groups */
} prcred_t;

The array of associated supplementary groups in pr_groups is of variable length; the cred file contains all of the supplementary groups. pr_ngroups indicates the number of supplementary groups. (See also the PCSCRED control operation.)

sigact
Contains an array of sigaction structures describing the current dispositions of all signals associated with the traced process (see sigaction(2)). Signal numbers are displaced by 1 from array indices, so that the action for signal number n appears in position n-1 of the array.
auxv | Contains the initial values of the process’s aux vector in an array of auxv_t structures (see <sys/auxv.h>). The values are those that were passed by the operating system as startup information to the dynamic linker.

ldt | This file exists only on IA based machines. It is non-empty only if the process has established a local descriptor table (LDT). If non-empty, the file contains the array of currently active LDT entries in an array of elements of type struct ssd, defined in <sys/sysi86.h>, one element for each active LDT entry.

map | Contains information about the virtual address map of the process. The file contains an array of prmap structures, each of which describes a contiguous virtual address region in the address space of the traced process:

```c
typedef struct prmap {
    uintptr_t pr_vaddr; /* virtual address of mapping */
    size_t pr_size; /* size of mapping in bytes */
    char pr_mapname[PRMAPSZ]; /* name in /proc/pid/object */
    offset_t pr_offset; /* offset into mapped object, if any */
    int pr_mflags; /* protection and attribute flags */
    int pr_pagesize; /* pagesize for this mapping in bytes */
    int pr_shmid; /* SysV shared memory identifier */
} prmap_t;
```

pr_vaddr is the virtual address of the mapping within the traced process and pr_size is its size in bytes. pr_mapname, if it does not contain a null string, contains the name of a file in the object directory (see below) that can be opened read-only to obtain a file descriptor for the mapped file associated with the mapping. This enables a debugger to find object file symbol tables without having to know the real path names of the executable file and shared libraries of the process. pr_offset is the 64-bit offset within the mapped file (if any) to which the virtual address is mapped.

pr_mflags is a bit-mask of protection and attribute flags:

- MA_READ: mapping is readable by the traced process.
- MA_WRITE: mapping is writable by the traced process.
- MA_EXEC: mapping is executable by the traced process.
- MA_SHARED: mapping changes are shared by the mapped object.
- MA_ISM: mapping is intimate shared memory (shared MMU resources).

A contiguous area of the address space having the same underlying mapped object may appear as multiple mappings due to varying read, write, and execute attributes. The underlying mapped object does not change over the range of a single mapping. An I/O operation to a mapping marked MA_SHARED fails if applied at a virtual address not corresponding to a valid page in the underlying mapped object. A write to a MA_SHARED mapping that is not marked MA_WRITE fails. Reads and writes to private mappings always succeed. Reads and writes to unmapped addresses fail.

pr_pagesize is the page size for the mapping, currently always the system pagesize.
### pr_shmid

`pr_shmid` is the shared memory identifier, if any, for the mapping. Its value is \(-1\) if the mapping is not System V shared memory. See `shmget(2)`.

### rmap

Contains information about the reserved address ranges of the process. The file contains an array of `prmap` structures, as defined above for the `map` file. Each structure describes a contiguous virtual address region in the address space of the traced process that is reserved by the system in the sense that an ` mmap(2)` system call that does not specify `MAP_FIXED` will not use any part of it for the new mapping. Examples of such reservations include the address ranges reserved for the process stack and the individual thread stacks of a multi-threaded process.

### cwd

A symbolic link to the process's current working directory (see `chdir(2)`). A `readlink(2)` of `/proc/pid/cwd` yields a null string. However, it can be opened, listed, and searched as a directory and can be the target of `chdir(2)`.

### root

A symbolic link to the process's root directory. `/proc/pid/root` can differ from the system root directory if the process or one of its ancestors executed `chroot(2)` as super-user. It has the same semantics as `/proc/pid/cwd`.

### fd

A directory containing references to the open files of the process. Each entry is a decimal number corresponding to an open file descriptor in the process.

If an entry refers to a regular file, it can be opened with normal file system semantics but, to ensure that the controlling process cannot gain greater access than the controlled process, with no file access modes other than its read/write open modes in the controlled process. If an entry refers to a directory, it appears as a symbolic link and can be accessed with the same semantics as `/proc/pid/cwd`. An attempt to open any other type of entry fails with EACCES.

### object

A directory containing read-only files with names corresponding to the `pr_mapname` entries in the `map` and `pagedata` files. Opening such a file yields a file descriptor for the underlying mapped file associated with an address-space mapping in the process. The file name `a.out` appears in the directory as an alias for the process's executable file.

The `object` directory makes it possible for a controlling process to gain access to the object file and any shared libraries (and consequently the symbol tables) without having to know the actual path names of the executable files.

### pagedata

Opening the page data file enables tracking of address space references and modifications on a per-page basis.

A `read(2)` of the page data file descriptor returns structured page data and atomically clears the page data maintained for the file by the system. That is to say, each read returns data collected since the last read; the first read returns data collected since the file was opened. When the call completes, the read buffer contains the following structure as its header and thereafter contains a number of section header structures and associated byte arrays that must be accessed by walking linearly through the buffer.
typedef struct prpageheader {
    timestruc_t pr_tstamp; /* real time stamp, time of read() */
    ulong_t pr_nmap; /* number of address space mappings */
    ulong_t pr_npaged; /* total number of pages */
} prpageheader_t;

The header is followed by pr_nmap prasmap structures and associated data arrays.

typedef struct prasmap {
    uintptr_t pr_vaddr; /* virtual address of mapping */
    ulong_t pr_npage; /* number of pages in mapping */
    char pr_mapname[PRMAPSZ]; /* name in /proc/pid/object */
    offset_t pr_offset; /* offset into mapped object, if any */
    int pr_mflags; /* protection and attribute flags */
    int pr_pagesize; /* pagesize for this mapping in bytes */
    int pr_shmid; /* SysV shared memory identifier */
} prasmap_t;

Each section header is followed by pr_npaged bytes, one byte for each page in the mapping, plus 0-7 null bytes at the end so that the next prasmap structure begins on an eight-byte aligned boundary. Each data byte may contain these flags:

PG_REFERENCED page has been referenced.

PG_MODIFIED page has been modified.

If the read buffer is not large enough to contain all of the page data, the read fails with E2BIG and the page data is not cleared. The required size of the read buffer can be determined through fstat(2). Application of lseek(2) to the page data file descriptor is ineffective; every read starts from the beginning of the file. Closing the page data file descriptor terminates the system overhead associated with collecting the data.

More than one page data file descriptor for the same process can be opened, up to a system-imposed limit per traced process. A read of one does not affect the data being collected by the system for the others. An open of the page data file will fail with ENOMEM if the system-imposed limit would be exceeded.

watch Contains an array of prwatch structures, one for each watched area established by the PCWATCH control operation. See PCWATCH for details.

usage Contains process usage information described by a prusage structure which contains at least the following fields:

typedef struct prusage {
    id_t pr_lwpid; /* lwp id. 0: process or defunct */
    int pr_count; /* number of contributing lwps */
    timestruc_t pr_tstamp; /* real time stamp, time of read() */
    timestruc_t pr_create; /* process/lwp creation time stamp */
    timestruc_t pr_term; /* process/lwp termination time stamp */
    timestruc_t pr_utime; /* total lwp real (elapsed) time */
    timestruc_t pr_utime; /* user level CPU time */
    timestruc_t pr_stime; /* system call CPU time */
    timestruc_t pr_stime; /* other system trap CPU time */
} prusage_t;
If microstate accounting has not been enabled for the process (see the PR_MSACCT flag for the PCSET operation, below), the usage file contains only an estimate of times spent in the various states. The usage file is accessible after a process becomes a zombie.

**lstatus**
Contains a prheader structure followed by an array of lwpstatus structures, one for each lwp in the process (see also /proc/pid/lwp/lwpid/lwpstatus, below). The prheader structure describes the number and size of the array entries that follow.

```c
typedef struct prheader {
    long pr_nent;    /* number of entries */
    size_t pr_entsize; /* size of each entry, in bytes */
} prheader_t;
```

The lwpstatus structure may grow by the addition of elements at the end in future releases of the system. Programs must use pr_entsize in the file header to index through the array. These comments apply to all /proc files that include a prheader structure (lpsinfo and lusage, below).

**lpsinfo**
Contains a prheader structure followed by an array of lwpsinfo structures, one for each lwp in the process. (See also /proc/pid/lwp/lwpid/lwpsinfo, below.)

**lusage**
Contains a prheader structure followed by an array of prusage structures, one for each lwp in the process plus an additional element at the beginning that contains the summation over all defunct lwps (lwps that once existed but no longer exist in the process). Excluding the pr_lwpid, pr_tstamp, pr_create, and pr_term entries, the entry-by-entry summation over all these structures is the definition of the process usage information obtained from the usage file. (See also /proc/pid/lwp/lwpid/lwpusage, below.)

**lwp**
A directory containing entries each of which names an lwp within the process. These entries are themselves directories containing additional files as described below.
A given directory `/proc/pid/lwp/lwpid` contains the following entries:

- **lwpstatus**: Lwp-specific state information. This file contains the `lwpstatus` structure for the specific lwp as described above for the representative lwp in the process's `status` file.

- **lwpsinfo**: Lwp-specific `ps(1)` information. This file contains the `lwpsinfo` structure for the specific lwp as described above for the representative lwp in the process's `psinfo` file.

- **lwpusage**: This file contains the `prusage` structure for the specific lwp as described above for the process's `usage` file.

- **gwindows**: This file exists only on SPARC based machines. If it is non-empty, it contains a `gwindows_t` structure, defined in `<sys/regset.h>`, with the values of those SPARC register windows that could not be stored on the stack when the lwp stopped. Conditions under which register windows are not stored on the stack are: the stack pointer refers to nonexistent process memory or the stack pointer is improperly aligned. If the lwp is not stopped or if there are no register windows that could not be stored on the stack, the file is empty (the usual case).

- **xregs**: Extra state registers. The extra state register set is architecture dependent; this file is empty if the system does not support extra state registers. If the file is non-empty, it contains an architecture dependent structure of type `prxregset_t`, defined in `<procfs.h>`, with the values of the lwp's extra state registers. If the lwp is not stopped, all register values are undefined. See also the PCSXREG control operation, below.

- **asrs**: This file exists only for 64-bit SPARC V9 processes. It contains an `asrset_t` structure, defined in `<sys/regset.h>`, containing the values of the lwp's platform-dependent ancillary state registers. If the lwp is not stopped, all register values are undefined. See also the PCSASRS control operation, below.

**CONTROL MESSAGES**

Process state changes are effected through messages written to a process's `ctl` file or to an individual lwp's `lwpctl` file. All control messages consist of a `long` that names the specific operation followed by additional data containing the operand, if any.

Multiple control messages may be combined in a single `write(2)` (or `writev(2)`) to a control file, but no partial writes are permitted. That is, each control message, operation code plus operand, if any, must be presented in its entirety to the `write(2)` and not in pieces over several system calls. If a control operation fails, no subsequent operations contained in the same `write(2)` are attempted.

Descriptions of the allowable control messages follow. In all cases, writing a message to a control file for a process or lwp that has terminated elicits the error `ENOENT`. 
When applied to the process control file, PCSTOP directs all lwps to stop and waits for them to stop, PCDSTOP directs all lwps to stop without waiting for them to stop, and PCWSTOP simply waits for all lwps to stop. When applied to an lwp control file, PCSTOP directs the specific lwp to stop and waits until it has stopped, PCDSTOP directs the specific lwp to stop without waiting for it to stop, and PCWSTOP simply waits for the specific lwp to stop. When applied to an lwp control file, PCSTOP and PCWSTOP complete when the lwp stops on an event of interest, immediately if already so stopped; when applied to the process control file, they complete when every lwp has stopped either on an event of interest or on a PR_SUSPENDED stop.

PCTWSTOP is identical to PCWSTOP except that it enables the operation to time out, to avoid waiting forever for a process or lwp that may never stop on an event of interest. PCTWSTOP takes a long operand specifying a number of milliseconds; the wait will terminate successfully after the specified number of milliseconds even if the process or lwp has not stopped; a timeout value of zero makes the operation identical to PCWSTOP.

An “event of interest” is either a PR_REQUESTED stop or a stop that has been specified in the process’s tracing flags (set by PCSTRACE, PCSFAULT, PSENTRY, and PCSEXIT). PR_JOBCONTROL and PR_SUSPENDED stops are specifically not events of interest. (An lwp may stop twice due to a stop signal, first showing PR_SIGNALLED if the signal is traced and again showing PR_JOBCONTROL if the lwp is set running without clearing the signal.) If PCSTOP or PCDSTOP is applied to an lwp that is stopped, but not on an event of interest, the stop directive takes effect when the lwp is restarted by the competing mechanism. At that time, the lwp enters a PR_REQUESTED stop before executing any user-level code.

A write of a control message that blocks is interruptible by a signal so that, for example, an alarm(2) can be set to avoid waiting forever for a process or lwp that may never stop on an event of interest. If PCSTOP is interrupted, the lwp stop directives remain in effect even though the write(2) returns an error. (Use of PCTWSTOP with a non-zero timeout is recommended over PCWSTOP with an alarm(2).)

A system process (indicated by the PR_ISSYS flag) never executes at user level, has no user-level address space visible through /proc, and cannot be stopped. Applying one of these operations to a system process or any of its lwps elicits the error EBUSY.

Make an lwp runnable again after a stop. This operation takes a long operand containing zero or more of the following flags:

PRCSIG clears the current signal, if any (see PCCSIG).
PRCFault clears the current fault, if any (see PCCFAULT).
PRSTEP directs the lwp to execute a single machine instruction. On completion of the instruction, a trace trap occurs. If FLTTRACE is being traced, the lwp stops; otherwise, it is sent SIGTRAP. If SIGTRAP is being traced and is not blocked, the lwp stops. When
the lwp stops on an event of interest, the single-step directive is cancelled, even if the stop occurs before the instruction is executed. This operation requires hardware and operating system support and may not be implemented on all processors. It is implemented on SPARC and IA based machines.

PRSABORT is meaningful only if the lwp is in a PR_SYSENTRY stop or is marked PR_ASLEEP; it instructs the lwp to abort execution of the system call (see PCSENTRY and PCSEXIT).

PRSTOP directs the lwp to stop again as soon as possible after resuming execution (see PCDSTOP). In particular, if the lwp is stopped on PR_SIGNALLED or PR_FAULTED, the next stop will show PR_REQUESTED, no other stop will have intervened, and the lwp will not have executed any user-level code.

When applied to an lwp control file, PCRUN clears any outstanding directed-stop request and makes the specific lwp runnable. The operation fails with EBUSY if the specific lwp is not stopped on an event of interest or has not been directed to stop or if the agent lwp exists and this is not the agent lwp (see PCAGENT).

When applied to the process control file, a representative lwp is chosen for the operation as described for /proc/pid/status. The operation fails with EBUSY if the representative lwp is not stopped on an event of interest or has not been directed to stop or if the agent lwp exists. If PRSTEP or PRSTOP was requested, the representative lwp is made runnable and its outstanding directed-stop request is cleared; otherwise all outstanding directed-stop requests are cleared and, if it was stopped on an event of interest, the representative lwp is marked PR_REQUESTED. If, as a consequence, all lwps are in the PR_REQUESTED or PR_SUSPENDED stop state, all lwps showing PR_REQUESTED are made runnable.

PCSTRACE Define a set of signals to be traced in the process. The receipt of one of these signals by an lwp causes the lwp to stop. The set of signals is defined using an operand sigset_t contained in the control message. Receipt of SIGKILL cannot be traced; if specified, it is silently ignored.

If a signal that is included in an lwp’s held signal set (the signal mask) is sent to the lwp, the signal is not received and does not cause a stop until it is removed from the held signal set, either by the lwp itself or by setting the held signal set with PCSHOLD.

PCCSIG The current signal, if any, is cleared from the specific or representative lwp.

PCSSIG The current signal and its associated signal information for the specific or representative lwp are set according to the contents of the operand siginfo structure (see <sys/siginfo.h>). If the specified signal number is zero, the current signal is cleared. The semantics of this operation are different from those of kill(2) in that the
signal is delivered to the lwp immediately after execution is resumed (even if it is being blocked) and an additional PR_SIGNALLED stop does not intervene even if the signal is traced. Setting the current signal to SIGKILL terminates the process immediately.

**PCKILL**  
If applied to the process control file, a signal is sent to the process with semantics identical to those of `kill(2)`. If applied to an lwp control file, a directed signal is sent to the specific lwp. The signal is named in a long operand contained in the message. Sending SIGKILL terminates the process immediately.

**PCUNKILL**  
A signal is deleted, that is, it is removed from the set of pending signals. If applied to the process control file, the signal is deleted from the process’s pending signals. If applied to an lwp control file, the signal is deleted from the lwp’s pending signals. The current signal (if any) is unaffected. The signal is named in a long operand in the control message. It is an error (EINVAL) to attempt to delete SIGKILL.

**PCSHOLD**  
Set the set of held signals for the specific or representative lwp (signals whose delivery will be blocked if sent to the lwp). The set of signals is specified with a sigset_t operand. SIGKILL and SIGSTOP cannot be held; if specified, they are silently ignored.

**PCSFAULT**  
Define a set of hardware faults to be traced in the process. On incurring one of these faults, an lwp stops. The set is defined via the operand fltset_t structure. Fault names are defined in `<sys/fault.h>` and include the following. Some of these may not occur on all processors; there may be processor-specific faults in addition to these.

- FLTILL: illegal instruction
- FLT_PRIV: privileged instruction
- FLT_BPT: breakpoint trap
- FLTTRACE: trace trap (single-step)
- FLT_WATCH: watchpoint trap
- FLT_ACCESS: memory access fault (bus error)
- FLT_BOUNDS: memory bounds violation
- FLT_OVF: integer overflow
- FLT_ZDIV: integer zero divide
- FLT_FPE: floating-point exception
- FLT_STACK: unrecoverable stack fault
- FLT_PAGE: recoverable page fault
When not traced, a fault normally results in the posting of a signal to the lwp that incurred the fault. If an lwp stops on a fault, the signal is posted to the lwp when execution is resumed unless the fault is cleared by PCCFAULT or by the PRCFAULT option of PCRUN. FLTPAGE is an exception; no signal is posted. The pr_info field in the lwpstatus structure identifies the signal to be sent and contains machine-specific information about the fault.

**PCCFAULT**

The current fault, if any, is cleared; the associated signal will not be sent to the specific or representative lwp.

**PCSENTRY**

These control operations instruct the process’s lwps to stop on entry to or exit from specified system calls. The set of system calls to be traced is defined via an operand sysset_t structure.

When entry to a system call is being traced, an lwp stops after having begun the call to the system but before the system call arguments have been fetched from the lwp. When exit from a system call is being traced, an lwp stops on completion of the system call just prior to checking for signals and returning to user level. At this point, all return values have been stored into the lwp’s registers.

If an lwp is stopped on entry to a system call (PR_SYSENTRY) or when sleeping in an interruptible system call (PR_ASLEEP is set), it may be instructed to go directly to system call exit by specifying the PRSABORT flag in a PCRUN control message. Unless exit from the system call is being traced, the lwp returns to user level showing EINTR.

**PCSEXIT**

**PCWATCH**

Set or clear a watched area in the controlled process from a prwatch structure operand:

```c
typedef struct prwatch {
    uintptr_t pr_vaddr; /* virtual address of watched area */
    size_t pr_size; /* size of watched area in bytes */
    int pr_wflags; /* watch type flags */
} prwatch_t;
```

pr_vaddr specifies the virtual address of an area of memory to be watched in the controlled process. pr_size specifies the size of the area, in bytes. pr_wflags specifies the type of memory access to be monitored as a bit-mask of the following flags:

- **WA_READ** read access
- **WA_WRITE** write access
- **WA_EXEC** execution access
- **WA_TRAPAFTER** trap after the instruction completes

If pr_wflags is non-empty, a watched area is established for the virtual address range specified by pr_vaddr and pr_size. If pr_wflags is empty, any previously-established watched area starting at the specified virtual address is cleared; pr_size is ignored.
A watchpoint is triggered when an lwp in the traced process makes a memory reference that covers at least one byte of a watched area and the memory reference is as specified in pr_wflags. When an lwp triggers a watchpoint, it incurs a watchpoint trap. If FLTWATCH is being traced, the lwp stops; otherwise, it is sent a SIGTRAP signal; if SIGTRAP is being traced and is not blocked, the lwp stops.

The watchpoint trap occurs before the instruction completes unless WA_TRAPAFTER was specified, in which case it occurs after the instruction completes. If it occurs before completion, the memory is not modified. If it occurs after completion, the memory is modified (if the access is a write access).

pr_info in the lwpstatus structure contains information pertinent to the watchpoint trap. In particular, the si_addr field contains the virtual address of the memory reference that triggered the watchpoint, and the si_code field contains one of TRAP_RWATCH, TRAP_WWATCH, or TRAP_XWATCH, indicating read, write, or execute access, respectively. The si_trapafter field is zero unless WA_TRAPAFTER is in effect for this watched area; non-zero indicates that the current instruction is not the instruction that incurred the watchpoint trap. The si_pc field contains the virtual address of the instruction that incurred the trap.

A watchpoint trap may be triggered while executing a system call that makes reference to the traced process’s memory. The lwp that is executing the system call incurs the watchpoint trap while still in the system call. If it stops as a result, the lwpstatus structure contains the system call number and its arguments. If the lwp does not stop, or if it is set running again without clearing the signal or fault, the system call fails with EFAULT. If WA_TRAPAFTER was specified, the memory reference will have completed and the memory will have been modified (if the access was a write access) when the watchpoint trap occurs.

If more than one of WA_READ, WA_WRITE, and WA_EXEC is specified for a watched area, and a single instruction incurs more than one of the specified types, only one is reported when the watchpoint trap occurs. The precedence is WA_EXEC, WA_READ, WA_WRITE (WA_EXEC and WA_READ take precedence over WA_WRITE), unless WA_TRAPAFTER was specified, in which case it is WA_WRITE, WA_READ, WA_EXEC (WA_WRITE takes precedence).

PCWATCH fails with EINVAL if an attempt is made to specify overlapping watched areas or if pr_wflags contains flags other than those specified above. It fails with ENOMEM if an attempt is made to establish more watched areas than the system can support (the system can support thousands).

The child of a vfork(2) borrows the parent’s address space. When a vfork(2) is executed by a traced process, all watched areas established for the parent are suspended until the child terminates or performs an exec(2). Any watched areas established independently in the child are cancelled when the parent resumes after the child’s termination or exec(2). PCWATCH fails with EBUSY if applied to the parent of a vfork(2) before the child has terminated or performed an exec(2). The PR_VFORKP flag is set in the pstatus structure for such a parent process.
Certain accesses of the traced process’s address space by the operating system are immune to watchpoints. The initial construction of a signal stack frame when a signal is delivered to an lwp will not trigger a watchpoint trap even if the new frame covers watched areas of the stack. Once the signal handler is entered, watchpoint traps occur normally. On SPARC based machines, register window overflow and underflow will not trigger watchpoint traps, even if the register window save areas cover watched areas of the stack.

Watched areas are not inherited by child processes, even if the traced process’s inherit-on-fork mode, PR_FORK, is set (see PCSET, below). All watched areas are cancelled when the traced process performs a successful exec(2).

**PCSET PCUNSET**

PCSET sets one or more modes of operation for the traced process. PCUNSET unsets these modes. The modes to be set or unset are specified by flags in an operand long in the control message:

- **PR_FORK** (inherit-on-fork): When set, the process’s tracing flags and its inherit-on-fork mode are inherited by the child of a fork(2), fork1(2), or vfork(2). When unset, child processes start with all tracing flags cleared.

- **PR_RLC** (run-on-last-close): When set and the last writable /proc file descriptor referring to the traced process or any of its lwps is closed, all of the process’s tracing flags and watched areas are cleared, any outstanding stop directives are canceled, and if any lwps are stopped on events of interest, they are set running as though PCRUN had been applied to them. When unset, the process’s tracing flags and watched areas are retained and lwps are not set running on last close.

- **PR_KLC** (kill-on-last-close): When set and the last writable /proc file descriptor referring to the traced process or any of its lwps is closed, the process is terminated with SIGKILL.

- **PR_ASYNC** (asynchronous-stop): When set, a stop on an event of interest by one lwp does not directly affect any other lwp in the process. When unset and an lwp stops on an event of interest other than PR_REQUESTED, all other lwps in the process are directed to stop.

- **PR_MSACCT** (microstate accounting): When set, microstate accounting is enabled for the process. This allows the usage file to contain accurate values for the times the lwps spent in their various processing states. When unset (the default), the overhead of microstate accounting is avoided and the usage file can only contain an estimate of times spent in the various states.

- **PR_MSFORK** (inherit microstate accounting): When set, and microstate accounting is enabled for the process, microstate accounting will be enabled for future child processes. When unset, child processes start with microstate accounting disabled.
(breakpoint trap pc adjustment): On IA based machines, a breakpoint trap leaves the program counter (the EIP) referring to the breakpointed instruction plus one byte. When PR_BPTADJ is set, the system will adjust the program counter back to the location of the breakpointed instruction when the lwp stops on a breakpoint. This flag has no effect on SPARC based machines, where breakpoint traps leave the program counter referring to the breakpointed instruction.

PR_PTRACE (ptrace-compatibility): When set, a stop on an event of interest by the traced process is reported to the parent of the traced process via wait(2), SIGTRAP is sent to the traced process when it executes a successful exec(2), setuid/setgid flags are not honored for execs performed by the traced process, any exec of an object file that the traced process cannot read fails, and the process dies when its parent dies. This mode is deprecated; it is provided only to allow ptrace(2) to be implemented as a library function using /proc.

It is an error (EINVAL) to specify flags other than those described above or to apply these operations to a system process. The current modes are reported in the pr_flags field of /proc/pid/status and /proc/pid/lwp/lwpstatus.

PCSREG Set the general registers for the specific or representative lwp according to the operand prgregset_t structure.

On SPARC based systems, only the condition-code bits of the processor-status register (R_PSR) of SPARC V8 (32-bit) processes can be modified by PCSREG. Other privileged registers cannot be modified at all.

On IA based systems, only certain bits of the flags register (EFL) can be modified by PCSREG: these include the condition codes, direction-bit, and overflow-bit.

PCSREG fails with EBUSY if the lwp is not stopped on an event of interest.

PCSVADDR Set the address at which execution will resume for the specific or representative lwp from the operand long. On SPARC based systems, both %pc and %npc are set, with %npc set to the instruction following the virtual address. On IA based systems, only %eip is set. PCSVADDR fails with EBUSY if the lwp is not stopped on an event of interest.

PCSFPREG Set the floating-point registers for the specific or representative lwp according to the operand prfpregset_t structure. An error (EINVAL) is returned if the system does not support floating-point operations (no floating-point hardware and the system does not emulate floating-point machine instructions). PCSFPREG fails with EBUSY if the lwp is not stopped on an event of interest.
Set the extra state registers for the specific or representative lwp according to the architecture-dependent operand prxregset_t structure. An error (EINVAL) is returned if the system does not support extra state registers. PCSXREG fails with EBUSY if the lwp is not stopped on an event of interest.

PCSASRS

Set the ancillary state registers for the specific or representative lwp according to the SPARC V9 platform-dependent operand asrset_t structure. An error (EINVAL) is returned if either the target process or the controlling process is not a 64-bit SPARC V9 process. Most of the ancillary state registers are privileged registers that cannot be modified. Only those that can be modified are set; all others are silently ignored. PCSASRS fails with EBUSY if the lwp is not stopped on an event of interest.

PCAGENT

Create an agent lwp in the controlled process with register values from the operand prregset_t structure (see PCSREG, above). The agent lwp is created in the stopped state showing PR_REQUESTED and with its held signal set (the signal mask) having all signals except SIGKILL and SIGSTOP blocked.

The PCAGENT operation fails with EBUSY unless the process is fully stopped via /proc, that is, unless all of the lwps in the process are stopped either on events of interest or on PR_SUSPENDED, or are stopped on PR_JOBCONTROL and have been directed to stop via PCSTOP. It fails with EBUSY if an agent lwp already exists. It fails with ENOMEM if system resources for creating new lwps have been exhausted.

Any PCRUN operation applied to the process control file or to the control file of an lwp other than the agent lwp fails with EBUSY as long as the agent lwp exists. The agent lwp must be caused to terminate by executing the _lwp_exit(2) system call before the process can be restarted.

Once the agent lwp is created, its lwp-ID can be found by reading the process status file. To facilitate opening the agent lwp’s control and status files, the directory name /proc/pid/lwp/agent is accepted for lookup operations as an invisible alias for /proc/pid/lwp/lwpid, lwpid being the lwp-ID of the agent lwp (invisible in the sense that the name “agent” does not appear in a directory listing of /proc/pid/lwp obtained from ls(1), getdents(2), or readdir(3C)).

The purpose of the agent lwp is to perform operations in the controlled process on behalf of the controlling process: to gather information not directly available via /proc files, or in general to make the process change state in ways not directly available via /proc control operations. To make use of an agent lwp, the controlling process must be capable of making it execute system calls (specifically, the _lwp_exit(2) system call). The register values given to the agent lwp on creation are typically the registers of the representative lwp, so that the agent lwp can use its stack.

The agent lwp is not allowed to execute any variation of the fork(2), exec(2), or _lwp_create(2) system calls. Attempts to do so yield ENOTSUP to the agent lwp.

PCREAD, PCWRITE

Read or write the target process’s address space via a priovvec structure operand:
typedef struct priovec {
    void *pio_base; /* buffer in controlling process */
    size_t pio_len; /* size of read/write request in bytes */
    off_t pio_offset; /* virtual address in target process */
} priovec_t;

These operations have the same effect as `pread(2)` and `pwrite(2)`, respectively, of the target process’s address space file. The difference is that more than one PCREAD or PCWRITE control operation can be written to the control file at once, and they can be interspersed with other control operations in a single write to the control file. This is useful, for example, when planting many breakpoint instructions in the process’s address space, or when stepping over a breakpointed instruction. Unlike `pread(2)` and `pwrite(2)`, no provision is made for partial reads or writes; if the operation cannot be performed completely, it fails with EIO.

**PCNICE**
The traced process’s `nice(2)` value is incremented by the amount in the operand `long`. Only the super-user may better a process’s priority in this way, but any user may lower the priority. This operation is not meaningful for all scheduling classes.

**PCSCRED**
Set the target process credentials to the values contained in the `prcred_t` structure operand (see `/proc/pid/cred`). The effective, real, and saved user-IDs and group-IDs of the target process are set. The target process’s supplementary groups are not changed; the `pr_ngroups` and `pr_groups` members of the structure operand are ignored. Only the super-user may perform this operation; for all others it fails with EPERM.

**PROGRAMMING NOTES**
For security reasons, except for the `psinfo`, `usage`, `lpsinfo`, `lusage`, `lwpsinfo`, and `lwpsusage` files, which are world-readable, and except for the super-user, an open of a `/proc` file fails unless both the user-ID and group-ID of the caller match those of the traced process and the process’s object file is readable by the caller. Except for the world-readable files just mentioned, files corresponding to setuid and setgid processes can be opened only by the super-user.

Even if held by the super-user, an open process or lwp file descriptor (other than file descriptors for the world-readable files) becomes invalid if the traced process performs an `exec(2)` of a setuid/setgid object file or an object file that the traced process cannot read. Any operation performed on an invalid file descriptor, except `close(2)`, fails with EAGAIN. In this situation, if any tracing flags are set and the process or any lwp file descriptor is open for writing, the process will have been directed to stop and its run-on-last-close flag will have been set (see PCBSET). This enables a controlling process (if it has permission) to reopen the `/proc` files to get new valid file descriptors, close the invalid file descriptors, unset the run-on-last-close flag (if desired), and proceed. Just closing the invalid file descriptors causes the traced process to resume execution with all tracing flags cleared. Any process not currently open for writing via `/proc`, but that has left-over tracing flags from a previous open, and that executes a setuid/setgid or unreadable object file, will not be stopped but will have all its tracing flags cleared.
To wait for one or more of a set of processes or lwps to stop or terminate, /proc file descriptors (other than those obtained by opening the cwd or root directories or by opening files in the fd or object directories) can be used in a poll(2) system call. When requested and returned, either of the polling events POLLPRI or POLLWRNORM indicates that the process or lwp stopped on an event of interest. Although they cannot be requested, the polling events POLLHUP, POLLERR, and POLLNVAL may be returned. POLLHUP indicates that the process or lwp has terminated. POLLERR indicates that the file descriptor has become invalid. POLLNVAL is returned immediately if POLLPRI or POLLWRNORM is requested on a file descriptor referring to a system process (see PCSTOP). The requested events may be empty to wait simply for termination.

FILES

/proc
directory (list of processes)

/proc/pid
specific process directory

/proc/self
alias for a process’s own directory

/proc/pid/as
address space file

/proc/pid/ctl
process control file

/proc/pid/status
process status

/proc/pid/lstatus
array of lwp status structs

/proc/pid/psinfo
process ps(1) info

/proc/pid/lpsinfo
array of lwp ps(1) info structs

/proc/pid/map
address space map

/proc/pid/rmap
reserved address map

/proc/pid/cred
process credentials

/proc/pid/sigact
process signal actions

/proc/pid/auxv
process aux vector
/proc/pid/ldt
  process LDT (IA only)

/proc/pid/usage
  process usage

/proc/pid/lusage
  array of lwp usage structs

/proc/pid/pagedata
  process page data

/proc/pid/watch
  active watchpoints

/proc/pid/cwd
  symlink to the current working directory

/proc/pid/root
  symlink to the root directory

/proc/pid/fd
  directory (list of open files)

/proc/pid/fd/*
  aliases for process's open files

/proc/pid/object
  directory (list of mapped files)

/proc/pid/object/a.out
  alias for process's executable file

/proc/pid/object/*
  aliases for other mapped files

/proc/pid/lwp
  directory (list of lwps)

/proc/pid/lwp/lwpid
  specific lwp directory

/proc/pid/lwp/agent
  alias for the agent lwp directory

/proc/pid/lwp/lwpid/lwpctl
  lwp control file

/proc/pid/lwp/lwpid/lwpstatus
  lwp status

/proc/pid/lwp/lwpid/lwpsinfo
  lwp ps(1) info

/proc/pid/lwp/lwpid/lwpusage
  lwp usage
Errors that can occur in addition to the errors normally associated with file system access:

**ENOENT** The traced process or lwp has terminated after being opened.

**EIO** A `write(2)` was attempted at an illegal address in the traced process.

**EBUSY** `PCSTOP`, `PCDSTOP`, `PCWSTOP`, or `PCTWSTOP` was applied to a system process; an exclusive `open(2)` was attempted on a `/proc` file for a process already open for writing; `PCRUN`, `PCSREG`, `PCSVADDR`, `PCSSPREG`, or `PCSXREG` was applied to a process or lwp not stopped on an event of interest; an attempt was made to mount `/proc` when it was already mounted; `PCAGENT` was applied to a process that was not fully stopped or that already had an agent lwp.

**EPERM** Someone other than the super-user issued the `PCSCRED` operation; someone other than the super-user attempted to better a process’s priority by applying `PCNICE`.

**ENOSYS** An attempt was made to perform an unsupported operation (such as `creat(2)`, `link(2)`, or `unlink(2)`) on an entry in `/proc`.

**EINVAL** In general, this means that some invalid argument was supplied to a system call. A non-exhaustive list of conditions eliciting this error includes: a control message operation code is undefined; an out-of-range signal number was specified with `PCSSIG`, `PCKILL`, or `PCUNKILL`; `SIGKILL` was specified with `PCUNKILL`; `PCSFREG` was applied to a system that does not support floating-point operations; `PCSXREG` was applied to a system that does not support extra state registers.

**ENOMEM** The system-imposed limit on the number of page data descriptors was reached on an open of `/proc/pid/pagedata`; an
attempt was made with PCWATCH to establish more watched areas than the system can support; the PCAGENT operation was issued when the system was out of resources for creating lwps.

E2BIG Data to be returned in a read(2) of the page data file exceeds the size of the read buffer provided by the caller.

EINTR A signal was received by the controlling process while waiting for the traced process or lwp to stop via PCSTOP, PCWSTOP, or PCTWSTOP.

EAGAIN The traced process has performed an exec(2) of a setuid/setgid object file or of an object file that it cannot read; all further operations on the process or lwp file descriptor (except close(2)) elicit this error.

EOVERFLOW A 32-bit controlling process attempted to read or write the as file or attempted to read the map, rmap, or pagedata file of a 64-bit target process. A 32-bit controlling process attempted to apply one of the control operations PCSREG, PCSXREG, PCSVADDR, PCWATCH, PCAGENT, PCREAD, PCWRITE to a 64-bit target process.

NOTES Descriptions of structures in this document include only interesting structure elements, not filler and padding fields, and may show elements out of order for descriptive clarity. The actual structure definitions are contained in <procfs.h>.

BUGS Because the old ioctl(2)-based version of /proc is currently supported for binary compatibility with old applications, the top-level directory for a process, /proc/pid, is not world-readable, but it is world-searchable. Thus, anyone can open /proc/pid/psinfo even though ls(1) applied to /proc/pid will fail for anyone but the owner or the super-user. Support for the old ioctl(2)-based version of /proc will be dropped in a future release, at which time the top-level directory for a process will be made world-readable.

On SPARC based machines, the types gregset_t and fpregset_t defined in <sys/regset.h> are similar to but not the same as the types prgregset_t and prfpregset_t defined in <procfs.h>.
prof_attr(4)

NAME
prof_attr – profile description database

SYNOPSIS
/etc/security/prof_attr

DESCRIPTION
/etc/security/prof_attr is a local source for execution profile names,
descriptions, and other attributes of execution profiles. The prof_attr file can be
used with other profile sources, including the prof_attr NIS map and NIS+ table.
Programs use the getprofattr(3SECDB) routines to gain access to this information.

The search order for multiple prof_attr sources is specified in the
/etc/nsswitch.conf file, as described in the nsswitch.conf(4) man page.

An execution profile is a mechanism used to bundle together the commands and
authorizations needed to perform a specific function. Each entry in the prof_attr
database consists of one line of text containing five fields separated by colons (:).
Line continuations using the backslash (\) character are permitted. The format of each
entry is:

profname:res1:res2:desc:attr

profname
The name of the profile. Profile names are case-sensitive.

res1
Reserved for future use.

res2
Reserved for future use.

desc
A long description. This field should explain the purpose of the
profile, including what type of user would be interested in using it.
The long description should be suitable for displaying in the help
text of an application.

attr
An optional list of semicolon-separated (;) key-value pairs that
describe the security attributes to apply to the object upon
execution. Zero or more keys may be specified. There are two valid
keys, help and auths.

help is assigned the name of a file ending in .htm or .html.

auths specifies a comma-separated (,) list of authorization names
chosen from those names defined in the auth_attr(4) database.
Authorization names may be specified using the asterisk (*)
character as a wildcard. For example, solaris.printer.*
would mean all of Sun's authorizations for printing.

EXAMPLES

EXAMPLE 1 Allowing execution of all commands

The following entry allows the user to execute all commands:

All: : :Use this profile to give a :help=All.html
EXAMPLE 1 Allowing execution of all commands (Continued)

EXAMPLE 2 Consulting the local prof_attr file first

With the following nsswitch.conf entry, the local prof_attr file is consulted before the NIS+ table:

```
prof_attr: files nisplus
```

FILES
/etc/nsswitch.conf
/etc/security/prof_attr

NOTES
When deciding which authorization source to use (see DESCRIPTION), keep in mind that NIS+ provides stronger authentication than NIS.

The root user is usually defined in local databases because root needs to be able to log in and do system maintenance in single-user mode and at other times when the network name service databases are not available. So that the profile definitions for root can be located at such times, root’s profiles should be defined in the local prof_attr file, and the order shown in the example nsswitch.conf(4) file entry under EXAMPLES is highly recommended.

Because the list of legal keys is likely to expand, any code that parses this database must be written to ignore unknown key-value pairs without error. When any new keywords are created, the names should be prefixed with a unique string, such as the company’s stock symbol, to avoid potential naming conflicts.

Each application has its own requirements for whether the help value must be a relative pathname ending with a filename or the name of a file. The only known requirement is for the name of a file.

The following characters are used in describing the database format and must be escaped with a backslash if used as data: colon (:), semicolon (;), equals (=), and backslash (\).

SEE ALSO auths(1), profiles(1), getauthattr(3SECDB), getprofattr(3SECDB), getuserattr(3SECDB), auth_attr(4), exec_attr(4), user_attr(4)
NAME  profile – setting up an environment for user at login time

SYNOPSIS  
/etc/profile

$HOME/.profile

DESCRIPTION  All users who have the shell, sh(1), as their login command have the commands in these files executed as part of their login sequence.

/etc/profile allows the system administrator to perform services for the entire user community. Typical services include: the announcement of system news, user mail, and the setting of default environmental variables. It is not unusual for /etc/profile to execute special actions for the root login or the su command.

The file $HOME/.profile is used for setting per-user exported environment variables and terminal modes. The following example is typical (except for the comments):

```
# Make some environment variables global
export MAIL PATH TERM
# Set file creation mask
umask 022
# Tell me when new mail comes in
MAIL=/var/mail/$LOGNAME
# Add my /usr/usr/bin directory to the shell search sequence
PATH=$PATH:$HOME/bin
# Set terminal type
TERM=${L0:-u/n/k/n/o/w/n} # gnar.invalid
while :
do
   if [ -f ${TERMINFO:-/usr/share/lib/terminfo}/?$TERM ]
      then break
   elif [ -f /usr/share/lib/terminfo/?$TERM ]
      then break
   else echo "invalid term $TERM" 1>&2
      fi
   echo "terminal: \c"
   read TERM
done
# Initialize the terminal and set tabs
# Set the erase character to backspace
stty erase "^H" echo
```

FILES  
$HOME/.profile  user-specific environment
/etc/profile  system-wide environment

SEE ALSO  env(1), login(1), mail(1), sh(1), stty(1), tput(1), su(1M), terminfo(4), environ(5), term(5)

OpenWindows Advanced User’s Guide

NOTES  Care must be taken in providing system-wide services in /etc/profile. Personal .profile files are better for serving all but the most global needs.
The project file is a local source of project information. The project file can be used in conjunction with other project sources, including the NIS maps project.byname and project.bynumber and the LDAP database project. Programs use the getprojent(3EXACCT) routines to access this information.

The project file contains a one-line entry for each project recognized by the system, of the form:

```
projname:projid:comment:user-list:group-list:attributes
```

where the fields are defined as:

- **projname**: The name of the project. Allowable project names must begin with a letter, and may be composed of any letter or digit and the underscore character. The period (`.`) is reserved for projects with special meaning to the operating system.

- **projid**: The project's unique numerical ID (PROJID) within the system.

- **comment**: The project's description.

- **user-list**: A comma-separated list of users allowed in the project.

- **group-list**: A comma-separated list of groups of users allowed in the project.

- **attributes**: A semicolon-separated list of name value pairs. Each pair has the following format:

  ```
  name[=value]
  ```

  where `name` is the arbitrary string specifying the key's name and `value` is the optional key value. An explanation of the valid name-value pair syntax is provided in the usage section of this page. The expected most frequent use of the attribute field is for the specification of resource controls.

The maximum value of the projid field is MAXPROJID.

Malformed entries cause routines that read this file to halt, in which case project assignments specified further along are never made. Blank lines are treated as malformed entries in the project file, and will cause getprojent(3EXACCT) and derived interfaces to fail.

### Examples

**Example 1: Sample project File**

The following is a sample project file:

```
system:0:System:::
user.root:1:Super-User:::
noproject:2:No Project:::
default:3::
```
EXAMPLE 1 Sample project File  (Continued)

group.staff:10:::
beatles:100:The Beatles:john,paul,george,ringo::task.max-lwps=
   (privileged,100,signal=SIGTERM), (privileged,110,deny)

Note that the line break in the line that begins with beatles is not valid in a project file. It is shown here only to allow the example to display on a printed or displayed page. Each entry must be on one and only one line.

An example project entry for nsswitch.conf(4) is:

project: files nis

With these entries, the project beatles will have members john, paul, george, and ringo, and all projects listed in the NIS project table are effectively incorporated after the entry for beatles.

The beatles project has two values set on the task.max-lwps resource control. When a task in the beatles project requests (via one of its member processes) its 100th and 110th LWPs, an action associated with the encountered threshold triggers. Upon the request for the 100th LWP, the process making the request is sent the signal SIGTERM and is granted the request for an additional lightweight process (LWP). At this point, the threshold for 110 LWPs becomes the active threshold. When a request for the 110th LWP in the task is made, the requesting process is denied the request—no LWP will be created. Since the 110th LWP is never granted, the threshold remains active, and all subsequent requests for an 110th LWP will fail. (If LWPs are given up, then subsequent requests will succeed, unless they would take the total number of LWPs across the task over 110.)

USAGE  The project database offers a reasonably flexible attribute mechanism in the final name-value pair field. Name-value pairs are separated from one another with the semicolon (;) character. The name is in turn distinguished from the (optional) value by the equals (=) character. The value field can contain multiple values separated by the comma (,) character, with grouping support (into further values lists) by parentheses. Each of these values can be composed of the upper and lower case alphabetic characters, the digits ‘0’ through ‘9’, and the punctuation characters hyphen (-), plus (+), period (.), slash (/), and underscore (_). Example resource control value specifications are provided in EXAMPLES, above, and on the getprojent(3EXACCT) manual page.

SEE ALSO  newtask(1), projects(1), getprojent(3EXACCT), unistd(3HEAD), nsswitch.conf(4)
The protocols file is a local source of information regarding the known protocols used in the DARPA Internet. The protocols file can be used in conjunction with or instead of other protocols sources, including the NIS maps "protocols.byname" and "protocols.bynumber" and the NIS+ table "protocols". Programs use the getprotobyname(3SOCKET) routine to access this information.

The protocols file has one line for each protocol. The line has the following format:

```
official-protocol-name protocol-number aliases
```

Items are separated by any number of blanks and/or TAB characters. A ‘#’ indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file. Protocol names may contain any printable character other than a field delimiter, NEWLINE, or comment character.

### EXAMPLE 1 A Sample Database

The following is a sample database:

```
# Internet (IP) protocols
#
ip  0 IP       # internet protocol, pseudo protocol number
icmp 1 ICMP    # internet control message protocol
ggp 3 GGP      # gateway-gateway protocol
tcp  6 TCP     # transmission control protocol
eggp  8 EGP     # exterior gateway protocol
pup  12 PUP     # PARC universal packet protocol
udp  17 UDP     # user datagram protocol

# Internet (IPv6) extension headers
#
hopopt 0 HOPOPT  # Hop-by-hop options for IPv6
ipv6 41 IPv6     # IPv6 in IP encapsulation
ipv6-route 43 IPv6-Route # Routing header for IPv6
ipv6-frag 44 IPv6-Frag # Fragment header for IPv6
esp 50 ESP      # Encap Security Payload for IPv6
ah  51 AH       # Authentication Header for IPv6
ipv6-icmp 58 IPv6-ICMP # IPv6 internet control message protocol
ipv6-nonxt 59 IPv6-NoNxt # No next header extension header for IPv6
ipv6-opts 60 IPv6-Opts # Destination Options for IPv6
```

### FILES

```
/etc/nsswitch.conf  configuration file for name-service switch
```

### SEE ALSO

```
getprotobyname(3SOCKET), nsswitch.conf(4)
```
NOTES

/etc/inet/protocols is the official SVR4 name of the protocols file. The symbolic link /etc/protocols exists for BSD compatibility.
prototype is an ASCII file used to specify package information. Each entry in the file describes a single deliverable object. An object may be a data file, directory, source file, executable object, and so forth. This file is generated by the package developer.

Entries in a prototype file consist of several fields of information separated by white space. Comment lines begin with a “#” and are ignored. The fields are described below and must appear in the order shown.

- **part**
  An optional field designating the part number in which the object resides. A part is a collection of files and is the atomic unit by which a package is processed. A developer can choose criteria for grouping files into a part (for example, based on class). If this field is not used, part 1 is assumed.

- **ftype**
  A one-character field that indicates the file type. Valid values are:
  - b: block special device
  - c: character special device
  - d: directory
  - e: a file to be edited upon installation or removal (may be shared by several packages)
  - f: a standard executable or data file
  - i: installation script or information file
  - l: linked file
  - p: named pipe
  - s: symbolic link
  - v: volatile file (one whose contents are expected to change, like a log file)
  - x: an exclusive directory accessible only by this package

- **class**
  The installation class to which the file belongs. This name must contain only alphanumeric characters and be no longer than 12 characters. The field is not specified for installation scripts. (admin and all classes beginning with capital letters are reserved class names.)

- **pathname**
  The pathname where the file will reside on the target machine, for example, /usr/bin/mail or bin/ras/proc. Relative pathnames (those that do not begin with a slash) indicate that the file is relocatable. The form
  
  path1=path2

---

**NAME**

prototype – package information file

**DESCRIPTION**

prototype is an ASCII file used to specify package information. Each entry in the file describes a single deliverable object. An object may be a data file, directory, source file, executable object, and so forth. This file is generated by the package developer.

Entries in a prototype file consist of several fields of information separated by white space. Comment lines begin with a “#” and are ignored. The fields are described below and must appear in the order shown.

- **part**
  An optional field designating the part number in which the object resides. A part is a collection of files and is the atomic unit by which a package is processed. A developer can choose criteria for grouping files into a part (for example, based on class). If this field is not used, part 1 is assumed.

- **ftype**
  A one-character field that indicates the file type. Valid values are:
  - b: block special device
  - c: character special device
  - d: directory
  - e: a file to be edited upon installation or removal (may be shared by several packages)
  - f: a standard executable or data file
  - i: installation script or information file
  - l: linked file
  - p: named pipe
  - s: symbolic link
  - v: volatile file (one whose contents are expected to change, like a log file)
  - x: an exclusive directory accessible only by this package

- **class**
  The installation class to which the file belongs. This name must contain only alphanumeric characters and be no longer than 12 characters. The field is not specified for installation scripts. (admin and all classes beginning with capital letters are reserved class names.)

- **pathname**
  The pathname where the file will reside on the target machine, for example, /usr/bin/mail or bin/ras/proc. Relative pathnames (those that do not begin with a slash) indicate that the file is relocatable. The form
  
  path1=path2

---
may be used for two purposes: to define a link and to define local pathnames.

For linked files, *path1* indicates the destination of the link and *path2* indicates the source file. (This format is mandatory for linked files.)

For local pathnames, *path1* indicates the pathname an object should have on the machine where the entry is to be installed and *path2* indicates either a relative or fixed pathname to a file on the host machine which contains the actual contents.

A pathname may contain a variable specification of the form `$variable`. If *variable* begins with a lower case letter, it is a build variable. If *variable* begins with an upper case letter, it is an install variable. Build variables are bound at build time. If an install variable is known at build time, its definition is inserted into the *pkginfo*(4) file so that it will be available at install time. If an install variable is not known at build time, it will be bound at install time.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>major</strong></td>
<td>The major device number. The field is only specified for block or character special devices.</td>
</tr>
<tr>
<td><strong>minor</strong></td>
<td>The minor device number. The field is only specified for block or character special devices.</td>
</tr>
<tr>
<td><strong>mode</strong></td>
<td>The octal mode of the file (for example, 0664). A question mark (?) indicates that the mode will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked files or packaging information files.</td>
</tr>
<tr>
<td><strong>owner</strong></td>
<td>The owner of the file (for example, <em>bin</em> or <em>root</em>). The field is limited to 14 characters in length. A question mark (?) indicates that the owner will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked files or packaging information files.</td>
</tr>
</tbody>
</table>

The mode can be a variable specification of the form `$variable`. If *variable* begins with a lower case letter, it is a build variable. If *variable* begins with an upper case letter, it is an install variable. Build variables are bound at build time. If an install variable is known at build time, its definition is inserted into the *pkginfo*(4) file so that it will be available at install time. If an install variable is not known at build time, it will be bound at install time.

The owner can be a variable specification of the form `$variable`. If *variable* begins with a lower case letter, it is a build variable. If *variable* begins with an upper case letter, it is an install variable. Build variables are bound at build time. If an install variable is
known at build time, its definition is inserted into the \texttt{pkginfo(4)} file so that it will be available at install time. If an install variable is not known at build time, it will be bound at install time.

\begin{verbatim}

group
\end{verbatim}

The group to which the file belongs (for example, \texttt{bin} or \texttt{sys}). The field is limited to 14 characters in length. A question mark (?) indicates that the group will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked files or packaging information files.

The group can be a variable specification of the form \texttt{$\backslash$variable}. If \texttt{variable} begins with a lower case letter, it is a build variable. If \texttt{variable} begins with an upper case letter, it is an install variable. Build variables are bound at build time. If an install variable is known at build time, its definition is inserted into the \texttt{pkginfo(4)} file so that it will be available at install time. If an install variable is not known at build time, it will be bound at install time.

An exclamation point (!) at the beginning of a line indicates that the line contains a command. These commands are used to incorporate files in other directories, to locate objects on a host machine, and to set permanent defaults. The following commands are available:

\begin{verbatim}

search
\end{verbatim}

Specifies a list of directories (separated by white space) to search for when looking for file contents on the host machine. The base name of the \texttt{path} field is appended to each directory in the ordered list until the file is located. Searches are not recursive.

\begin{verbatim}

include
\end{verbatim}

Specifies a pathname which points to another prototype file to include. Note that \texttt{search} requests do not span \texttt{include} files.

\begin{verbatim}

default
\end{verbatim}

Specifies a list of attributes (mode, owner, and group) to be used by default if attribute information is not provided for prototype entries which require the information. The defaults do not apply to entries in \texttt{include} prototype files.

\begin{verbatim}

param=value
\end{verbatim}

Places the indicated parameter in the current environment. Spans to subsequent included prototype files.

The above commands may have variable substitutions embedded within them, as demonstrated in the two example prototype files below.

Before files are overwritten during installation, they are copied to a temporary path name. The exception to this rule is files whose mode includes execute permission, unless the file is editable (that is, \texttt{ftype} is \texttt{e}). For files which meet this exception, the existing version is linked to a temporary path name, and the original file is removed. This allows processes which are executing during installation to be overwritten.
EXAMPLE 1

Example 1:

```bash
PROJDIR=/usr/proj
BIN=$PROJDIR/bin
CFG=$PROJDIR/cfg
LIB=$PROJDIR/lib
HDRS=$PROJDIR/hdrs

search /usr/myname/usr/bin /usr/myname/src /usr/myname(hdrs
i pkginfo=/usr/myname/wrap/pkginfo
i depend=/usr/myname/wrap/depend
i version=/usr/myname/wrap/version
d none /usr/wrap 0755 root bin

search $BIN
f none /usr/wrap/bin/INSTALL 0755 root bin
f none /usr/wrap/bin/REMOVE 0755 root bin
f none /usr/wrap/bin/addpkg 0755 root bin

default 0755 root bin
f none /usr/wrap/bin/audit
f none /usr/wrap/bin/listpkg

# the following file starts out zero length but grows
v none /usr/wrap/logfile=/dev/null 0644 root bin

# the following specifies a link (dest=src)
l none /usr/wrap/src/addpkg=/usr/wrap/bin/rmpkg

search $SRC

f src /usr/wrap/src/INSTALL.sh
f src /usr/wrap/src/REMOVE.sh
f src /usr/wrap/src/audit.c
f src /usr/wrap/src/listpkg.c
f src /usr/wrap/src/pkgmk.c

default 644 root other
f src /usr/wrap/src/data 0755 root bin
d none /usr/wrap/save 0755 root bin
d none /usr/wrap/spool 0755 root bin
d none /usr/wrap/tmp 0755 root bin
d src /usr/wrap/src 0755 root bin
```

EXAMPLE 2

Example 2:

```bash
# this prototype is generated by `pkgproto` to refer
to all prototypes in my src directory
PROJDIR=/usr/dew/projx
#include $PROJDIR/src/cmd/prototype
#include $PROJDIR/src/cmd/audmerg/protofile
#include $PROJDIR/src/lib/proto

SEE ALSO

pkgs(1), pkginfo(4)

Application Packaging Developer’s Guide
prototype(4)

NOTES  Normally, if a file is defined in the prototype file but does not exist, that file is created at the time of package installation. However, if the file pathname includes a directory that does not exist, the file will not be created. For example, if the prototype file has the following entry:

f none /usr/dev/bin/command

and that file does not exist, it will be created if the directory /usr/dev/bin already exists or if the prototype also has an entry defining the directory:

d none /usr/dev/bin
pseudo – configuration files for pseudo device drivers

DESCRIPTION

Pseudo devices are devices that are implemented entirely in software. Drivers for pseudo devices must provide driver configuration files to inform the system of each pseudo device that should be created.

Configuration files for pseudo device drivers must identify the parent driver explicitly as pseudo, and must create an integer property called instance which is unique to this entry in the configuration file.

Each entry in the configuration file creates a prototype devinfo node. Each node is assigned an instance number which is determined by the value of the instance property. This property is only applicable to children of the pseudo parent, and is required since pseudo devices have no hardware address from which to determine the instance number. See driver.conf(4) for further details of configuration file syntax.

EXAMPLE 1

A sample configuration file.

Here is a configuration file called ramdisk.conf for a pseudo device driver that implements a RAM disk. This file creates two nodes called "ramdisk". The first entry creates ramdisk node instance 0, and the second creates ramdisk node, instance 1, with the additional disk-size property set to 512.

# # Copyright (c) 1993, by Sun Microsystems, Inc. #
#ident "@(#)ramdisk.conf 1.3 93/06/04 SMI"
name="ramdisk" parent="pseudo" instance=0;
name="ramdisk" parent="pseudo" instance=1 disk-size=512;

SEE ALSO

driver.conf(4), ddi_prop_op(9F)

Writing Device Drivers
NAME    publickey – public key database
SYNOPSIS /etc/publickey
DESCRIPTION /etc/publickey is a local public key database that is used for secure RPC. The /etc/publickey file can be used in conjunction with or instead of other publickey databases, including the NIS publickey map and the NIS+ publickey map. Each entry in the database consists of a network user name (which may refer to either a user or a hostname), followed by the user’s public key (in hex notation), a colon, and then the user’s secret key encrypted with a password (also in hex notation).

The /etc/publickey file contains a default entry for nobody.

SEE ALSO chkey(1), newkey(1M), getpublickey(3NSL), nsswitch.conf(4)
The queuedefs file describes the characteristics of the queues managed by cron(1M). Each non-comment line in this file describes one queue. The format of the lines are as follows:

```
q.[njob][nicen][nwaitw]
```

The fields in this line are:

- **q**: The name of the queue. `a` is the default queue for jobs started by `at(1)`; `b` is the default queue for jobs started by `batch` (see `at(1)`); `c` is the default queue for jobs run from a `crontab(1)` file.

- **njob**: The maximum number of jobs that can be run simultaneously in that queue; if more than `njob` jobs are ready to run, only the first `njob` jobs will be run, and the others will be run as jobs that are currently running terminate. The default value is 100.

- **nice**: The `nice(1)` value to give to all jobs in that queue that are not run with a user ID of super-user. The default value is 2.

- **nwait**: The number of seconds to wait before rescheduling a job that was deferred because more than `njob` jobs were running in that job’s queue, or because the system-wide limit of jobs executing has been reached. The default value is 60.

Lines beginning with `#` are comments, and are ignored.

**EXAMPLE 1** A sample file.

```
# #
a.4j1n
b.2j2n90w
```

This file specifies that the a queue, for `at` jobs, can have up to 4 jobs running simultaneously; those jobs will be run with a `nice` value of 1. As no `nwait` value was given, if a job cannot be run because too many other jobs are running, `cron` will wait 60 seconds before trying again to run it.

The b queue, for `batch(1)` jobs, can have up to 2 jobs running simultaneously; those jobs will be run with a `nice(1)` value of 2. If a job cannot be run because too many other jobs are running, `cron(1M)` will wait 90 seconds before trying again to run it. All other queues can have up to 100 jobs running simultaneously; they will be run with a `nice` value of 2, and if a job cannot be run because too many other jobs are running, `cron` will wait 60 seconds before trying again to run it.
### queuedefs(4)

| FILES   | /etc/cron.d/queuedefs | queue description file for `at`, `batch`, and `cron`. |
| SEE ALSO | `at(1)`, `crontab(1)`, `nice(1)`, `cron(1M)` |
NAME
rcmscript – script interface specification for the Reconfiguration and Coordination Manager

SYNOPSIS
rcm_scriptname scriptinfo
rcm_scriptname register
rcm_scriptname resourceinfo resourcename
rcm_scriptname queryremove resourcename
rcm_scriptname preremove resourcename
rcm_scriptname postremove resourcename
rcm_scriptname undoremove resourcename

DESCRIPTION
Reconfiguration and Coordination Manager (RCM) is a framework designed to coordinate device consumers during Solaris Dynamic Reconfiguration (DR). The interfaces specified in this man page allow device consumers, such as application vendors or site administrators, to act before and after DR operations take place by providing RCM scripts. You can write your own RCM scripts to shut down your applications, or to cleanly release the devices from your applications during dynamic remove operations.

An RCM script is an executable perl script, a shell script or a binary. Perl is the recommended language. Each script is run in its own address space using the user-id of the script file owner.

An RCM script is invoked on demand in response to DR as follows:

```bash
<scriptname> <command> [args ...]
```

Every script must implement the following RCM commands:

- **scriptinfo**: Get script information.
- **register**: Register devices the script handles.
- **resourceinfo**: Get resource information.

A script might include some or all of the following commands:

- **queryremove**: Queries whether the resource can be released.
- **preremove**: Releases the resource.
- **postremove**: Provides post-resource removal notification.
- **undoremove**: Undo the actions done in preremove.

When a script’s register command is run, the script should supply, in return data, all resource names the script or its application handles that could potentially be removed by DR. A resource name refers to a name in /dev path name.
Below is a high-level overview of the sequence of script invocations that occurs when dynamic removal of a script’s registered resource is attempted. See the COMMANDS section for a detailed description of the commands.

1. Prior to removing the resource from the system during DR, the script’s queryremove command is run:
   
   ```bash
   <scriptname> queryremove <resourcename>
   ```

   The script should check for obvious reasons why the resource can not be removed from the perspective of its service or application.

2. If the script indicates that the resource can be removed in the queryremove command. The script’s preremove command is run:
   
   ```bash
   <scriptname> preremove <resourcename>
   ```

   The script releases the resource from the service or application represented by the script and prepares for the resource removal. Releasing the resource includes closing the resource if the resource is currently opened by its application.

3. The system then proceeds to remove the resource.

4. If the system has removed the resource successfully the script’s postremove command is run:
   
   ```bash
   <scriptname> postremove <resourcename>
   ```

   Otherwise the script’s undoremove command is run:
   
   ```bash
   <scriptname> undoremove <resourcename>
   ```

For any commands the script does not implement, it must exit with exit status of 2. RCM silently returns success for the script’s unimplemented commands.

A script performs the following basic steps:

- Takes RCM command and additional arguments from the command line and environment parameters.
- Processes the command.
- Writes the expected return data to stdout as `name=value` pairs delimited by newlines, where `name` is the name of the return data item that RCM expects and `value` is the value associated with the data item.

Environment

Initial environment of RCM scripts is set as follows:

- Process UID is set to the UID of the script.
- Process GID is set to the GID of the script.
- `PATH` variable is set to `/usr/sbin:/usr/bin`. 
Current working directory is set to:
- /var/run for scripts owned by root
- /tmp for scripts not owned by root

File descriptor 0 (stdin) is set to /dev/null

Environment variable RCM_ENV_DEBUG_LEVEL is set to the debug level. Logging is discussed below.

The following environment variables are also set where possible:
- LANG
- LC_COLLATE
- LC_CTYPE
- LC_MESSAGES
- LC_MONETARY
- LC_NUMERIC
- LC_TIME
- LC_ALL
- TZ See environ(5) for a description of these variables. See gettext(1) for details on retrieving localized messages.

All environment variable names beginning with RCM_ENV_ are reserved for use by the RCM.

The character encoding used by the RCM and RCM scripts to exchange RCM commands, environment parameters, and name-value pairs is ASCII unless the controlling environment variables are specified otherwise.

Commands

scriptinfo
The scriptinfo command is invoked to gather information about the script.

Return data:
If successful, the script must write the following name-value pairs to stdout and exit with status 0:
- rcm_script_version=1
- rcm_script_func_info=script_func_info
- rcm_cmd_timeout=command_timeout_value where script_func_info is a localized human-readable message describing the functionality of the script.

The RCM monitors the execution time of RCM commands by RCM scripts. command_timeout_value is the maximum time in seconds the script is expected to take to process any RCM command except the scriptinfo command itself. If an RCM script does not process the RCM command and exit within this time, RCM sends a SIGABRT signal to the script process. RCM then waits for a few seconds for the script to finish the processing of the current RCM command and exit. If the script does not exit within this time, RCM sends a SIGKILL signal to the script.
The `rcm_cmd_timeout` name-value pair is optional. It is only needed if the script is expected to take more than a few seconds to process any RCM command. Setting this name to a value of 0 (zero) disables the timer. If this name-value pair is not supplied, a default value is assigned by the RCM.

Upon failure, the script must specify the failure reason using the name-value pair `rcm_failure_reason` and exit with status 1.

**register**

The `register` command is invoked to allow a script to specify the resources that it or its application handles that could potentially be removed by DR. The script has to supply all its resource names to RCM using the name-value pair `rcm_resource_name`.

**Return Data:**

If successful, the script must write the following name-value pairs to stdout and exit with status 0:

```
rcm_resource_name=resourcename
rcm_resource_name=resourcename
...
```

where `resourcename` is the name of the resource the script is interested in.

Upon failure, the script must specify the failure reason using the name-value pair `rcm_failure_reason` and exit with status 1.

**resourceinfo resourcename**

The `resourceinfo` command is invoked to get the usage information about `resourcename`.

**Return Data:**

If successful, the script must write the following name-value pair to stdout and exit with status 0:

```
rcm_resource_usage_info=resource_usage
```

where `resource_usage` is a localized human readable message describing the usage of the resource by the script.

Upon failure, the script must specify the failure reason using the name-value pair `rcm_failure_reason` and exit with status 1.

**queryremove resourcename**

Prior to removing the resource from the system, the `queryremove` command is invoked to query the script to determine whether the script can release the given resource successfully from the service or application it represents. The script does not actually release the resource. The script might indicate that it is not able to release the resource if the resource is critical for its service or application.
Additional environment parameter:
RCM_ENV_FORCE
Can be one of:

FALSE
Normal request.

TRUE
Request is urgent. The script should check whether the resource can be released successfully by force, such as by using the force option to unmount a file system.

Return Data:
If the command succeeds, the script must return no data and exit with status 0.

If the script would not be able to release the resource, it must specify the reason using the name-value pair rcm_failure_reason and exit with status 3.

Upon any other failure, the script must specify the failure reason using the name-value pair rcm_failure_reason and exit with status 1.

**preremove resourcename**
The preremove command is invoked prior to an attempt to remove the given resourcename. In response to this command the script can either release the resource (including closing the device if the device is currently opened) from the service or application it represents or indicate that it cannot release the resource if the resource is critical for its service or application.

Additional environment parameter:
RCM_ENV_FORCE
Can be one of:

FALSE
Normal request.

TRUE
Request is urgent. The script should make extra effort to release the resource, such as by using the force option to unmount a file system.

Return Data:
If the command succeeds, the script must return no data and exit with status 0.

If the script cannot release the resource, it must specify the reason using the name-value pair rcm_failure_reason and exit with status 3.

Upon any other failure, the script must specify the failure reason using the name-value pair rcm_failure_reason and exit with status 1.

**postremove resourcename**
The postremove command is invoked after the given resourcename has been removed.
Return Data:
If the command succeeds, the script must return no data and exit with status 0.
Upon failure, the script must specify the failure reason using the name-value pair
rcm_failure_reason and exit with status 1.

undoresource resourcename

The undoresource command is invoked to undo what was done in the previous
preremove command for the given resourcename. The script can bring the state of the
resource to the same state it was in when the script received the preremove
command for that resource.

Return Data:
If the command succeeds, the script must return no data and exit with status 0.
Upon failure, the script must specify the failure reason using the name-value pair
rcm_failure_reason and exit with status 1.

Logging
A script must log all error and debug messages by writing to stdout the name-value
pairs listed below. The logged messages go to syslogd(1M) with the syslog facility
of LOG_DAEMON. See syslog.conf(4).

rcm_log_err=message Logs the message with the syslog level of LOG_ERR.

rcm_log_warn=message Logs the message with the syslog level of
LOG_WARNING.

rcm_log_info=message Logs the message with the syslog level of LOG_INFO.

rcm_log_debug=message Logs the message with the syslog level of LOG_DEBUG.

A script can use the environment variable RCM_ENV_DEBUG_LEVEL to control the
amount of information to log. RCM_ENV_DEBUG_LEVEL is a numeric value ranging
from 0 to 9, with 0 meaning log the least amount of information and 9 meaning log the
most.

Installing or
Removing RCM
Scripts
You must use the following format to name a script:

vendor, service

where vendor is the stock symbol (or any distinctive name) of the vendor providing the
script and service is the name of service the script represents.

You must be a superuser (root) to install or remove an RCM script.

Select one of the following directories where you want to place the script:

/etc/rcm/scripts
Scripts for specific systems

/usr/platform/`uname -i`/lib/rcm/scripts
Scripts for specific hardware implementation
Installing a Script

To install a script, copy the script to the appropriate directory from the list above, change the userid and the groupid of the script to the desired values, and send SIGHUP to `rcm_daemon`. For example:

```
# cp SUNW,sample.pl /usr/lib/rcm/scripts
# chown user[:group] /usr/lib/rcm/scripts/SUNW,sample.pl
# pkill -HUP -x -u root rcm_daemon
```

Removing a script

Remove the script from the appropriate directory from the list above and send SIGHUP to `rcm_daemon`. For example:

```
# rm /usr/lib/rcm/scripts/SUNW,sample.pl
# pkill -HUP -x -u root rcm_daemon
```

EXAMPLE 1 Site Customization RCM Script

```perl
#!/usr/bin/perl -w

# A sample site customization RCM script for a tape backup application.
# When the system attempts to remove a tape drive by DR the script
# does the following:
# - if the tape drive is not being used for backup, it allows the
#   DR to continue.
# - if the tape drive is being used for backup, and when DR is not forced
#   (RCM_ENV_FORCE=FALSE) it indicates that it cannot release the
#   tape drive with appropriate error message. When forced
#   (RCM_ENV_FORCE=TRUE) it kills the tape backup application in
#   order to allow the DR to continue.
#
# This script does not implement the postremove and undoremove commands
# since there is nothing to cleanup after DR remove operation is completed
# or failed. If any cleanup is needed after the DR removal completed,
# postremove command needs to implemented. If any cleanup is needed
# in the event of DR removal failure, undoremove command needs to be
# implemented.
#
use strict;

my ($cmd, %dispatch);

$cmd = shift (@ARGV);
```

File Formats 401
rcmscript(4)
EXAMPLE 1

Site Customization RCM Script

(Continued)

# dispatch table for RCM commands
%dispatch = (
"scriptinfo"
=>
\&do_scriptinfo,
"register"
=>
\&do_register,
"resourceinfo" =>
\&do_resourceinfo,
"queryremove"
=>
\&do_preremove,
"preremove"
=>
\&do_preremove
);
if (defined($dispatch{$cmd})) {
&{$dispatch{$cmd}};
} else {
exit (2);
}
sub do_scriptinfo
{
print "rcm_script_version=1\n";
print "rcm_script_func_info=Tape backup appl script for DR\n";
exit (0);
}
sub do_register
{
my ($dir, $f, $errmsg);
$dir = opendir(RMT, "/dev/rmt");
if (!$dir) {
$errmsg = "Unable to open /dev/rmt directory: $!";
print "rcm_failure_reason=$errmsg\n";
exit (1);
}
while ($f = readdir(RMT)) {
# ignore hidden files and multiple names for the same device
if (($f !~ /^\./) && ($f =~ /^[0-9]+$/)) {
print "rcm_resource_name=/dev/rmt/$f\n";
}
}
closedir(RMT);
exit (0);
}
sub do_resourceinfo
{
my ($rsrc, $unit);
$rsrc = shift(@ARGV);
if ($rsrc =~ /^\/dev\/rmt\/([0-9]+)$/) {
$unit = $1;
print "rcm_resource_usage_info=Backup Tape Unit Number $unit\n";
exit (0);
} else {

402

man pages section 4: File Formats • Last Revised 12 Feb 2001


EXAMPLE 1 Site Customization RCM Script  (Continued)

```perl
print "rcm_failure_reason=Unknown tape device!\n";
exit (1);
}
}

sub do_preremove
{
    my ($rsrc);

    $rsrc = shift(@ARGV);

    # check if backup application is using this resource
    # if (the backup application is not running on $rsrc) {
    #    allow the DR to continue
    #    exit (0);
    #}
    #
    # If RCM_ENV_FORCE is FALSE deny the operation.
    # If RCM_ENV_FORCE is TRUE kill the backup application in order
    # to allow the DR operation to proceed
    #
    if ($ENV{RCM_ENV_FORCE} eq 'TRUE') {
        if ($cmd eq 'preremove') {
            # kill the tape backup application
        }
        exit (0);
    } else {
        #
        # indicate that the tape drive can not be released
        # since the device is being used for backup by the
        # tape backup application
        #
        print "rcm_failure_reason=tape backup in progress pid=...\n";
        exit (3);
    }
}
```

EXIT STATUS

A script must exit with following exit status values:

0  Operation specified by the given RCM command has been executed successfully by the script. For queryremove command it also means that the script can successfully release the resource.

1  An error occurred while processing the RCM command. The script should provide the error message to RCM using the name-value pair rcm_failure_reason before exiting.

2  The script does not support the given RCM command. A script must exit with this status if it cannot understand the given RCM command.

3  Indicates that the script cannot release the resource for preremove and queryremove commands. The script should provide a message to RCM.
specifying the reason for not being able to release the resource using the name-value pair `rcm_failure_reason` before exiting.

**ERRORS**

If a script cannot successfully process an RCM command, it must supply to the RCM a message indicating the reason for failure by writing a name-value pair, in the form shown below, to stdout and exiting with the appropriate exit status.

```
rcm_failure_reason=failure_reason
```

where `failure_reason` is a localized human readable message describing the reason for failure of the RCM 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>Interface Stability</td>
<td>Evolving</td>
</tr>
</tbody>
</table>

**SEE ALSO**

`gettext(1), cfgadm(1M), cfgadm_scsi(1M), cfgadm_pci(1M), syslog(3C), signal(3HEAD), syslog.conf(4), attributes(5), environ(5)`

**NOTES**

RCM scripts are expected to properly handle all RCM commands that the script implements and to log all errors. Only root has permission to add or remove an RCM script. An ill-behaved RCM script can cause unexpected DR failures.

RCM commands are invoked only for the resources whose subsystems participate within the RCM framework. Currently, not all subsystems participate within the RCM framework.
NAME  remote – remote host description file
SYNOPSIS /etc/remote
DESCRIPTION The systems known by tip(1) and their attributes are stored in an ASCII file which is structured somewhat like the termcap file. Each line in the file provides a description for a single system. Fields are separated by a colon ‘:’. Lines ending in a ‘\’ character with an immediately following NEWLINE are continued on the next line.

The first entry is the name(s) of the host system. If there is more than one name for a system, the names are separated by vertical bars. After the name of the system comes the fields of the description. A field name followed by an ‘=’ sign indicates a string value follows. A field name followed by a ‘#’ sign indicates a following numeric value.

Entries named tipbaudrate are used as default entries by tip, as follows. When tip is invoked with only a phone number, it looks for an entry of the form tipbaudrate, where baudrate is the baud rate with which the connection is to be made. For example, if the connection is to be made at 300 baud, tip looks for an entry of the form tip300.

CAPABILITIES Capabilities are either strings (str), numbers (num), or boolean flags (bool). A string capability is specified by capability=value; for example, ‘dv=/dev/harris’. A numeric capability is specified by capability#value; for example, ‘xa#99’. A boolean capability is specified by simply listing the capability.

at (str) Auto call unit type. The following lists valid ‘at’ types and their corresponding hardware:

- biz31f  Bizcomp 1031, tone dialing
- biz31w  Bizcomp 1031, pulse dialing
- biz22f  Bizcomp 1022, tone dialing
- biz22w  Bizcomp 1022, pulse dialing
- df02    DEC DF02
- df03    DEC DF03
- ventel  Ventel 212+
- v3451   Vadic 3451 Modem
- v831    Vadic 831
- hayes   Any Hayes-compatible modem
- at      Any Hayes-compatible modem

br (num) The baud rate used in establishing a connection to the remote host. This is a decimal number. The default baud rate is 300 baud.
**remote(4)**

| cm | (str) An initial connection message to be sent to the remote host. For example, if a host is reached through a port selector, this might be set to the appropriate sequence required to switch to the host. |
| cu | (str) Call unit if making a phone call. Default is the same as the dv field. |
| db | (bool) Cause tip(1) to ignore the first hangup it sees. db (dialback) allows the user to remain in tip while the remote machine disconnects and places a call back to the local machine. For more information about dialback configuration, see *System Administration Guide, Volume 3* |
| di | (str) Disconnect message sent to the host when a disconnect is requested by the user. |
| du | (bool) This host is on a dial-up line. |
| dv | (str) Device(s) to open to establish a connection. If this file refers to a terminal line, tip attempts to perform an exclusive open on the device to insure only one user at a time has access to the port. |
| ec | (bool) Initialize the tip variable echocheck to on, so that tip will synchronize with the remote host during file transfer by waiting for the echo of the last character transmitted. |
| el | (str) Characters marking an end-of-line. The default is no characters. tip only recognizes ‘~’ escapes after one of the characters in el, or after a RETURN. |
| es | (str) The command prefix (escape) character for tip. |
| et | (num) Number of seconds to wait for an echo response when echo-check mode is on. This is a decimal number. The default value is 10 seconds. |
| ex | (str) Set of non-printable characters not to be discarded when scripting with beautification turned on. The default value is “\t\n\b\f”. |
| fo | (str) Character used to force literal data transmission. The default value is ‘\377’. |
| fs | (num) Frame size for transfers. The default frame size is equal to 1024. |
| hd | (bool) Initialize the tip variable halfduplex to on, so local echo should be performed. |
| hf | (bool) Initialize the tip variable hardwareflow to on, so hardware flow control is used. |
| ie | (str) Input end-of-file marks. The default is a null string (“”). |
| nb | (bool) Initialize the tip variable beautify to off, so that unprintable characters will not be discarded when scripting. |
| nt | (bool) Initialize the tip variable tandem to off, so that XON/XOFF flow control will not be used to throttle data from the remote host. |
nv  (bool) Initialize the tip variable verbose to off, so that verbose mode will be turned on.

oe  (str) Output end-of-file string. The default is a null string ("""). When tip is transferring a file, this string is sent at end-of-file.

pa  (str) The type of parity to use when sending data to the host. This may be one of even, odd, none, zero (always set bit 8 to 0), one (always set bit 8 to 1). The default is none.

pn  (str) Telephone number(s) for this host. If the telephone number field contains an '@' sign, tip searches the /etc/phones file for a list of telephone numbers — see phones(4). A '%' sign in the telephone number indicates a 5-second delay for the Ventel Modem.

For Hayes-compatible modems, if the telephone number starts with an 'S', the telephone number string will be sent to the modem without the "DT", which allows reconfiguration of the modem's S-registers and other parameters; for example, to disable auto-answer: "pn=S0=0DT5551234"; or to also restrict the modem to return only the basic result codes: "pn=S0=0X0DT5551234".

pr  (str) Character that indicates end-of-line on the remote host. The default value is '\n'.

ra  (bool) Initialize the tip variable raise to on, so that lower case letters are mapped to upper case before sending them to the remote host.

rc  (str) Character that toggles case-mapping mode. The default value is '\377'.

re  (str) The file in which to record session scripts. The default value is tip.record.

rw  (bool) Initialize the tip variable rawftp to on, so that all characters will be sent as is during file transfers.

sc  (bool) Initialize the tip variable script to on, so that everything transmitted by the remote host will be recorded.

tb  (bool) Initialize the tip variable tabexpand to on, so that tabs will be expanded to spaces during file transfers.

tc  (str) Indicates that the list of capabilities is continued in the named description. This is used primarily to share common capability information.

EXAMPLES

EXAMPLE 1 The Capability Continuation Feature

Here is a short example showing the use of the capability continuation feature:

UNIX-1200:
  :dv=/dev/cua0;el="D"O"C"S"Q"O@;du:at=ventel;ie=#$%;oe="D;br#1200:
arpavax|ax:\
  :pm=7654321%;tc=UNIX-1200
remote(4)

FILES

/etc/remote           remote host description file.
/etc/phones           remote host phone number database.

SEE ALSO

tip(1), phones(4)

System Administration Guide, Volume 3
**NAME**  
resolv.conf – resolver configuration file

**SYNOPSIS**  
/etc/resolv.conf

**DESCRIPTION**  
The resolver is a set of routines that provide access to the Internet Domain Name System. See resolver(3RESOLV). resolv.conf is a configuration file that contains the information that is read by the resolver routines the first time they are invoked by a process. The file is designed to be human readable and contains a list of keywords with values that provide various types of resolver information.

The resolv.conf file contains the following configuration directives:

<table>
<thead>
<tr>
<th>Directive</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nameserver</td>
<td>Specifies the Internet address in dot-notation format of a name server that the resolver is to query. Up to MAXNS name servers may be listed, one per keyword. See &lt;resolv.h&gt;. If there are multiple servers, the resolver library queries them in the order listed. If no name server entries are present, the resolver library queries the name server on the local machine. The resolver library follows the algorithm to try a name server until the query times out. It then tries the the name servers that follow, until each query times out. It repeats all the name servers until a maximum number of retries are made.</td>
</tr>
<tr>
<td>domain</td>
<td>Specifies the local domain name. Most queries for names within this domain can use short names relative to the local domain. If no domain entry is present, the domain is determined from sysinfo(2) or from gethostname(3C). (Everything after the first ‘.’ is presumed to be the domain name.) If the host name does not contain a domain part, the root domain is assumed. You can use the LOCALDOMAIN environment variable to override the domain name.</td>
</tr>
<tr>
<td>search</td>
<td>The search list for host name lookup. The search list is normally determined from the local domain name. By default, it contains only the local domain name. You can change the default behavior by listing the desired domain search path following the search keyword, with spaces or tabs separating the names. Most resolver queries will be attempted using each component of the search path in turn until a match is found. This process may be slow and will generate a lot of network traffic if the servers for the listed domains are not local. Queries will time out if no server is available for one of the domains.</td>
</tr>
</tbody>
</table>
The search list is currently limited to six domains and a total of 256 characters.

**sortlist**

Allows addresses returned by the libresolv-internal `gethostbyname()` to be sorted. A sortlist is specified by IP address netmask pairs. The netmask is optional and defaults to the natural netmask of the net. The IP address and optional network pairs are separated by slashes. Up to 10 pairs may be specified. For example:

```plaintext
sortlist 130.155.160.0/255.255.240.0 130.155.0.0
```

**options**

Allows certain internal resolver variables to be modified. The syntax is

```plaintext
options option ...
```

where option is one of the following:

**debug**

Sets `RES_DEBUG` in the `_res.options` field.

**ndots:** \( n \)

Sets a threshold floor for the number of dots which must appear in a name given to `res_query()` before an initial absolute (as-is) query is performed. See `resolver(3RESOLV)`. The default value for \( n \) is 1, which means that if there are any dots in a name, the name is tried first as an absolute name before any search list elements are appended to it.

**timeout:** \( n \)

**retrans:** \( n \)

Sets the amount of time the resolver will wait for a response from a remote name server before retrying the query by means of a different name server. Measured in seconds, the default is `RES_TIMEOUT`. See `<resolv.h>`. The `timeout` and `retrans` values are the starting point for an exponential back off procedure where the `timeout` is doubled for every retransmit attempt.

**attempts:** \( n \)

**retry:** \( n \)

Sets the number of times the resolver will send a query to its name servers before giving up and returning an error to the calling application. The default is `RES_DFLRETRY`. See `<resolv.h>`.
rotate
Sets RES_ROTATE in _res.options. The name servers are queried round-robin from among those listed. The query load is spread among all listed servers, rather than having all clients try the first listed server first every time.

no-check-names
Sets RES_NOCHECKNAME in _res.options. This disables the modern BIND checking of incoming host names and mail names for invalid characters such as underscore (_), non-ASCII, or control characters.

inet6
Sets RES_USE_INET6 in _res.options. In the Solaris BIND port, this has no effect on gethostbyname(3NSL). To retrieve IPv6 addresses or IPv4 addresses in mapped form, use getipnodebyname(3SOCKET) instead of setting inet6.

The domain and search keywords are mutually exclusive. If more than one instance of these keywords is present, the last instance takes precedence.

You can override the search keyword of the system resolv.conf file on a per-process basis by setting the environment variable LOCALDOMAIN to a space-separated list of search domains.

You can amend the options keyword of the system resolv.conf file on a per-process basis by setting the environment variable RES_OPTIONS to a space-separated list of resolver options.

The keyword and value must appear on a single line. Start the line with the keyword, for example, nameserver, followed by the value, separated by white space.

FILES
/etc/resolv.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>Interface Stability</td>
<td>Standard BIND 8.2.2</td>
</tr>
</tbody>
</table>

SEE ALSO
domainname(1M), in.named(1M), sysinfo(2), gethostbyname(3NSL), getipnodebyname(3SOCKET), gethostname(3C), resolver(3RESOLV)

The rmmount.conf file contains the rmmount(1M) configuration information. This file describes where to find shared objects that perform actions on file systems after identifying and mounting them. The rmmount.conf file is also used to share CD-ROM and floppy file systems. It can also direct the rmmount utility to run fsck on one or more file systems before mounting them, with the fsck command line options specified in rmmount.conf. The rmmount.conf file is also used to share CD-ROM, floppy, and other removable disk file systems.

Actions are executed in the order in which they appear in the configuration file. The action function can return either 1 or 0. If it returns 0, no further actions will be executed. This allows the function to control which applications are executed. For example, action_filemgr always returns 0 if the File Manager is running, thereby preventing subsequent actions from being executed.

To execute an action after a medium has been inserted and while the File Manager is not running, list the action after action_filemgr in the rmmount.conf file. To execute an action before the File Manager becomes aware of the medium, list the action before action_filemgr in the rmmount.conf file.

The syntax for the rmmount.conf file is as follows.

```markdown
# File system identification
ident filesystem_type shared_object media_type [media_type ...]

# Actions
action media_type shared_object args_to_so

# File system sharing
share media_or_file_system share_command_options

# Mount command options
mount media_or_file_system [file_system_spec] -o mount_command_options

# Optionally fsck command options
fsck media_type filesystem_type -o fsck_command_options
```

Explanations of the syntax for the File system identification fields are as follows.

- **filesystem_type**: An ASCII string used as the file system type flag of the mount command (see the -F option of mount(1M)). It is also used to match names passed to rmmount(1M) from Volume Management.

- **shared_object**: Programs that identify file systems and perform actions. This shared_object is found at /usr/lib/fs/filesystem_type/shared_object.
**media_type**

The type of medium where this file system resides. Legal values are *cdrom*, *floppy* and *rmdisk*.

Explanations of the syntax for the **Actions** fields are as follows.

**media_type**

Type of medium. This argument is passed in from Volume Management as `VOLUME_TYPE`.

**shared_object**

Programs that identify file systems and perform actions. If `shared_object` starts with ‘/’ (slash), the full path name is used; otherwise, `/usr/lib/rmmount` is prepended to the name.

**args_to_so**

Arguments passed to the `shared_object`. These arguments are passed in as an *argc* and *argv*.

The definition of the interface to **Actions** is located in `/usr/include/rmmount.h`.

Explanations of the syntax for the **File system sharing** fields are as follows.

**media_or_file_system**

Either the type of medium (CD-ROM or floppy) or the specific file system to share.

**share_command_options**

Options of the `share` command. See `share(1M)` for more information about these options.

Explanations of the syntax for the **Mount command options** fields are as follows:

**media_or_file_system**

Either the type of medium (CD-ROM or floppy) or the specific file system to share.

**file_system_spec**

Specifies one or more file systems to which this line applies. Defaults to "all" file system types.

**mount_command_options**

One or more options to be passed to the `mount` command. Multiple options require a space delimiter.

Explanations of the syntax for the **fsck command options** fields are as follows:

**media_type**

The type of removable medium. A Bourne shell regular expression that matches names of file system media whose aliases are listed under `/vol/dev/aliases`. Examples include `cdrom0`, `cdrom1`, `cdrom*`, `floppy0`, and `floppy1`, and `floppy*`.

**filesystem_type**

The type of file system, for example, *ufs* or *hsfs*, that resides on the medium specified in `media_type`.

**fsck_command_options**

One or more options to be passed to `fsck(1M)`. Multiple options must be separated by spaces.

The algorithm for the **fsck** configuration line is as follows:

1. The `fsck` configuration line tells `rmmount` to run `fsck` on `filesystem_type`, as described above. The `filesystem_type` must be correct for the `media_type` specified.
2. If `filesystem_type` is not present, `rmmount` runs `fsck` on all file systems on all media that match `media_type`.

3. If `rmmount.conf` contains no `fsck` configuration line or contains an `fsck` configuration line with a `media_type` that does not match a medium’s alias, `rmmount` does not run `fsck` on the removable medium’s file system, unless `mount` reports that the file system’s dirty bit is set.

**Default Values**

The following is an example of an `rmmount.conf` file.

```plaintext
# Removable Media Mounter configuration file.

# File system identification
ident hsfs ident_hsfs.so cdrom
ident ufs ident_ufs.so cdrom floppy rmscsi pcmem
ident pcfs ident_pcfs.so floppy rmscsi pcmem
ident udfs ident_udfs.so cdrom floppy

# Actions
action cdrom action_filemgr.so
action floppy action_filemgr.so
action rmscsi action_filemgr.so
```

**EXAMPLES**

**EXAMPLE 1** Sharing of various file systems.

The following examples show how various file systems are shared using the `share` syntax for the `rmmount.conf` file. These lines are added after the `Actions` entries.

- `share cdrom*`  
  Shares all CD-ROMs via NFS and applies no access restrictions.

- `share solaris_2.x*`  
  Shares CD-ROMs named `solaris_2.x*` with no access restrictions.

- `share cdrom* -o ro=engineering`  
  Shares all CD-ROMs via NFS but exports only to the “engineering” netgroup.

- `share solaris_2.x* -d distribution CD`  
  Shares CD-ROMs named `solaris_2.x*` with no access restrictions and with the description that it is a distribution CD-ROM.

- `share floppy0`  
  Shares any floppy inserted into floppy drive 0.

**EXAMPLE 2** Customizing mount operations

The following examples show how different `mount` options could be used to customize how `rmmount` mounts various media:

- `mount cdrom* hsfs -o nrr`  
  mounts all High Sierra CD-ROMs with the `nrr` (no Rock Ridge extensions) option (see `mount_hsfs(1M)`).
EXAMPLE 2 Customizing mount operations  (Continued)

mount floppy1 -o ro
  will always mount the second floppy disk read-only (for all file system types)

mount floppy1 -o ro foldcase
  will always mount the second floppy disk read-only (for all file system types) and pass the foldcase mount option

EXAMPLE 3 Telling rmmount to check file systems before mounting them

The following examples show how to tell rmmount to check file systems with fsck before mounting them, and how to specify the command line options to be used with fsck.

fsck floppy* ufs -o f
  Performs a full file system check on any UFS floppies, ignoring the clean flag, before mounting them.

fsck floppy* ufs -o p
  Uses the fsck p (preen) flag for all UFS floppies.

fsck cdrom* --o f
  Tells rmmount to run fsck before mounting any file system on CD-ROM.

SEE ALSO

volcancel(1), volcheck(1), volmissing(1), mount(1M), mount_hsfs(1M), rmmount(1M), share(1M), void(1M), void.conf(4), volfs(7FS)

NOTES

When using the mount options line, verify that the specified options will work with the specified file system types. The mount command will fail if an incorrect mount option/file system combination is specified. Multiple mount options require a space delimiter.
rmtab(4)

NAME  rmtab – remote mounted file system table

SYNOPSIS /etc/rmtab

DESCRIPTION rmtab contains a table of filesystems that are remotely mounted by NFS clients. This file is maintained by mountd(1M), the mount daemon. The data in this file should be obtained only from mountd(1M) using the MOUNTPROC_DUMP remote procedure call.

The file contains a line of information for each remotely mounted filesystem. There are a number of lines of the form:

hostname:fsname

The mount daemon adds an entry for any client that successfully executes a mount request and deletes the appropriate entries for an unmount request.

Lines beginning with a hash (‘#’) are commented out. These lines are removed from the file by mountd(1M) when it first starts up. Stale entries may accumulate for clients that crash without sending an unmount request.

FILES /etc/rmtab

SEE ALSO mountd(1M), showmount(1M)
The `rpc` file is a local source containing user readable names that can be used in place of RPC program numbers. The `rpc` file can be used in conjunction with or instead of other rpc sources, including the NIS maps `rpcbyname` and `rpc.bynumber` and the NIS+ table `rpc.`

The `rpc` file has one line for each RPC program name. The line has the following format:

```
name-of-the-RPC-program  RPC-program-number  aliases
```

Items are separated by any number of blanks and/or tab characters. A `#` indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file.

**EXAMPLE 1** RPC Database

Below is an example of an RPC database:

```
# # rpc
#
rpcbind 100000 portmap  sunrpc  portmapper
rusersd 100002  rusers
nfs 100003  nfsprog
mountd 100005  mount  showmount
wallld 100008  rwall  shutdown
sprayd 100012  spray
lockmgr 100020
nlockmgr 100021
status 100024
bootparam 100026
keyserv 100029  keyserver
```

**FILES** 

`/etc/nsswitch.conf`

**SEE ALSO**

`nsswitch.conf(4)`
NAME    rpld.conf – Remote Program Load (RPL) server configuration file

SYNOPSIS /etc/rpld.conf

DESCRIPTION The /etc/rpld.conf file contains the configuration information for operation of rpld, the RPL-based network boot server. It is a text file containing keyword-value pairs and comments. The keyword-value pairs specify the value to use for parameters used by the RPL server. Comments can be entered by starting the line using the # character. The user can add comments to the file for customized configurations. Alternate RPL server configuration files can be specified when running the RPL server by supplying a configuration file similar to the default configuration file.

Keywords All keywords are case-sensitive. Not all keywords must be present. (However, note that the end keyword at the end of the file must be present.) If a keyword is not present, internal defaults, which are the default values described here, will be used. Keyword-value pairs are specified by:

```
keyword = value
```

**DebugLevel** Specify the number of error, warning, and information messages to be generated while the RPL server is running. The valid range is 0-9. A value of 0 means no message at all, while a value of 9 will generate the most messages. The default is 0. Note that it is best to limit the value to 8 or below; use of level 9 may generate so many debug messages that the performance of the RPL server may be impacted.

**DebugDest** A numeric value specifying where to send the messages to:

```
0 = standard output
1 = syslogd
2 = log file
```

The default is 2.

**MaxClients** A numeric value specifying the maximum number of simultaneous network boot clients to be in service. A value of −1 means unlimited except where system resources is the limiting factor. Any positive value will set a limit on the number of clients to be in service at the same time unless system resource constraints come in before the limit. The default is −1.

**BackGround** A numeric value indicating whether the RPL server should run in the background or not. A 0 means run in the background and a 1 means do not run in the background. The difference is whether the server will relinquish the controlling terminal or not. The default is 1.

**FrameSize** The default size of data frames to be used to send bootfile data to the network boot clients. This size should not exceed the limits imposed by the underlying physical media. For
ethernet/802.3, the maximum physical frame size is 1500 octets. The default is 1500. Note that the protocol overhead of LLC1 and RPL is 32 octets, resulting in a maximum data length of 1468 octets.

LogFile
The log file to which messages will be sent if DebugDest is set to 2 (the default). The default file is var/spool/rpld.log.

StartDelay
The initial delay factor to use to control the speed of downloading. In the default mode of operation, the downloading process does not wait for a positive acknowledgment from the client before the next data frame is sent. In the case of a fast server and slow client, data overrun can result and requests for retransmission will be frequent. By using a delay factor, the speed of data transfer is controlled to avoid retransmission requests. Note that the unit of delay is machine dependent and bears no correlation with the actual time delayed.

DelayGran
Delay granularity. If the initial delay factor is not suitable and the rate of downloading is either too fast or too slow, retransmission requests from the clients will be used to adjust the delay factor either upward (to slow down the data rate) or downward (to speed up the data rate). The delay granularity is used as the delay delta for adjustment.

end
Keyword at the end of the file. It must be present.

FILES
/etc/rpld.conf
/usr/sbin/rpld

ATTRIBUTES
See 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
rpld(1M), attributes(5)
The process scheduler (or dispatcher) is the portion of the kernel that controls allocation of the CPU to processes. The scheduler supports the notion of scheduling classes where each class defines a scheduling policy, used to schedule processes within that class. Associated with each scheduling class is a set of priority queues on which ready to run processes are linked. These priority queues are mapped by the system configuration into a set of global scheduling priorities which are available to processes within the class. The dispatcher always selects for execution the process with the highest global scheduling priority in the system. The priority queues associated with a given class are viewed by that class as a contiguous set of priority levels numbered from 0 (lowest priority) to n (highest priority—a configuration dependent value). The set of global scheduling priorities that the queues for a given class are mapped into might not start at zero and might not be contiguous, depending on the configuration.

The real-time class maintains an in-core table, with an entry for each priority level, giving the properties of that level. This table is called the real-time dispatcher parameter table (rt_dptbl). The rt_dptbl consists of an array (config_rt_dptbl[]) of parameter structures (struct rtdpent_t), one for each of the n priority levels. The structure are accessed via a pointer, (rt_dptbl), to the array. The properties of a given priority level i are specified by the ith parameter structure in this array (rt_dptbl[i]).

A parameter structure consists of the following members. These are also described in the /usr/include/sys/rt.h header file.

- **rt_globpri**: The global scheduling priority associated with this priority level. The rt_globpri values cannot be changed with dispadmin(1M).
- **rt_quantum**: The length of the time quantum allocated to processes at this level in ticks (Hz). The time quantum value is only a default or starting value for processes at a particular level as the time quantum of a real-time process can be changed by the user with the priocntl command or the priocntl system call.

An administrator can affect the behavior of the real-time portion of the scheduler by reconfiguring the rt_dptbl. There are two methods available for doing this: reconfigure with a loadable module at boot-time or by using dispadmin(1M) at run-time.

The rt_dptbl can be reconfigured with a loadable module which contains a new real time dispatch table. The module containing the dispatch table is separate from the RT loadable module which contains the rest of the real time software. This is the only method that can be used to change the number of real time priority levels or the set of global scheduling priorities used by the real time class. The relevant procedure and source code is described in the EXAMPLES section.

---

**NAME**

rt_dptbl – real-time dispatcher parameter table

**DESCRIPTION**

The process scheduler (or dispatcher) is the portion of the kernel that controls allocation of the CPU to processes. The scheduler supports the notion of scheduling classes where each class defines a scheduling policy, used to schedule processes within that class. Associated with each scheduling class is a set of priority queues on which ready to run processes are linked. These priority queues are mapped by the system configuration into a set of global scheduling priorities which are available to processes within the class. The dispatcher always selects for execution the process with the highest global scheduling priority in the system. The priority queues associated with a given class are viewed by that class as a contiguous set of priority levels numbered from 0 (lowest priority) to n (highest priority—a configuration dependent value). The set of global scheduling priorities that the queues for a given class are mapped into might not start at zero and might not be contiguous, depending on the configuration.

The real-time class maintains an in-core table, with an entry for each priority level, giving the properties of that level. This table is called the real-time dispatcher parameter table (rt_dptbl). The rt_dptbl consists of an array (config_rt_dptbl[]) of parameter structures (struct rtdpent_t), one for each of the n priority levels. The structure are accessed via a pointer, (rt_dptbl), to the array. The properties of a given priority level i are specified by the ith parameter structure in this array (rt_dptbl[i]).

A parameter structure consists of the following members. These are also described in the /usr/include/sys/rt.h header file.

- **rt_globpri**: The global scheduling priority associated with this priority level. The rt_globpri values cannot be changed with dispadmin(1M).
- **rt_quantum**: The length of the time quantum allocated to processes at this level in ticks (Hz). The time quantum value is only a default or starting value for processes at a particular level as the time quantum of a real-time process can be changed by the user with the priocntl command or the priocntl system call.

An administrator can affect the behavior of the real-time portion of the scheduler by reconfiguring the rt_dptbl. There are two methods available for doing this: reconfigure with a loadable module at boot-time or by using dispadmin(1M) at run-time.

The rt_dptbl can be reconfigured with a loadable module which contains a new real time dispatch table. The module containing the dispatch table is separate from the RT loadable module which contains the rest of the real time software. This is the only method that can be used to change the number of real time priority levels or the set of global scheduling priorities used by the real time class. The relevant procedure and source code is described in the EXAMPLES section.
The `rt_quantum` values in the `rt_dptbl` can be examined and modified on a running system using the `dispadmin(1M)` command. Invoking `dispadmin` for the real-time class allows the administrator to retrieve the current `rt_dptbl` configuration from the kernel’s in-core table, or overwrite the in-core table with values from a configuration file. The configuration file used for input to `dispadmin` must conform to the specific format described below.

Blank lines are ignored and any part of a line to the right of a `#` symbol is treated as a comment. The first non-blank, non-comment line must indicate the resolution to be used for interpreting the time quantum values. The resolution is specified as

```
RES=res
```

where `res` is a positive integer between 1 and 1,000,000,000 inclusive and the resolution used is the reciprocal of `res` in seconds. (For example, `RES=1000` specifies millisecond resolution.) Although very fine (nanosecond) resolution may be specified, the time quantum lengths are rounded up to the next integral multiple of the system clock’s resolution.

The remaining lines in the file are used to specify the `rt_quantum` values for each of the real-time priority levels. The first line specifies the quantum for real-time level 0, the second line specifies the quantum for real-time level 1. There must be exactly one line for each configured real-time priority level. Each `rt_quantum` entry must be either a positive integer specifying the desired time quantum (in the resolution given by `res`), or the value `-2` indicating an infinite time quantum for that level.

**EXAMPLE 1** A Sample `dispadmin` Configuration File

The following excerpt from a `dispadmin` configuration file illustrates the format. Note that for each line specifying a time quantum there is a comment indicating the corresponding priority level. These level numbers indicate priority within the real-time class, and the mapping between these real-time priorities and the corresponding global scheduling priorities is determined by the configuration specified in the `RT_DPTBL` loadable module. The level numbers are strictly for the convenience of the administrator reading the file and, as with any comment, they are ignored by `dispadmin` on input. `dispadmin` assumes that the lines in the file are ordered by consecutive, increasing priority level (from 0 to the maximum configured real-time priority). The level numbers in the comments should normally agree with this ordering; if for some reason they don’t, however, `dispadmin` is unaffected.

```
# Real-Time Dispatcher Configuration File
RES=1000

# TIME QUANTUM PRIORITY
# (rt_quantum)LEVEL
100# 0
100# 1
100# 2
100# 3
100# 4
100# 5
```
EXAMPLE 1 A Sample dispadmin Configuration File (Continued)

```
90 # 6
90 # 7
.. ..
10# 58
10# 59
```

EXAMPLE 2 Replacing The rt_dptbl Loadable Module

In order to change the size of the real time dispatch table, the loadable module which contains the dispatch table information will have to be built. It is recommended that you save the existing module before using the following procedure.

1. Place the dispatch table code shown below in a file called rt_dptbl.c. An example of an rt_dptbl.c file follows.
2. Compile the code using the given compilation and link lines supplied.
   ```
   cc -c -0 -D_KERNEL rt_dptbl.c
   ld -r -o RT_DPTBL rt_dptbl.o
   ```
3. Copy the current dispatch table in /usr/kernel/sched to RT_DPTBL.bak.
4. Replace the current RT_DPTBL in /usr/kernel/sched.
5. You will have to make changes in the /etc/system file to reflect the changes to the sizes of the tables. See system(4). The rt_maxpri variable may need changing. The syntax for setting this is:
   ```
   set RT:rt_maxpri=(class-specific value for maximum real-time priority)
   ```
6. Reboot the system to use the new dispatch table.

Great care should be used in replacing the dispatch table using this method. If you don’t get it right, the system may not behave properly.

The following is an example of a rt_dptbl.c file used for building the new rt_dptbl.

```c
/* BEGIN rt_dptbl.c */
#include <sys/proc.h>
#include <sys/priocntl.h>
#include <sys/class.h>
#include <sys/disp.h>
#include <sys/rt.h>
#include <sys/rtpriocntl.h>
/*
 * This is the loadable module wrapper.
 */
#include <sys/modctl.h>
extern struct mod_ops mod_miscops;
/*
 * Module linkage information for the kernel.
 */
```
EXAMPLE 2 Replacing The rt_dptbl Loadable Module  (Continued)

*/
static struct modlmisc modlmisc = {
    &mod_miscops, "realtime dispatch table"
};
static struct modlinkage modlinkage = {
    MODREV_1, &modlmisc, 0
};
_init() {
    return (mod_install(&modlinkage));
}
_info (struct modinfo *modinfop) {
    return (mod_info(&modlinkage, modinfop));
}
rt_dpent_t config_rt_dptbl[] = {
    /* prilevel Time quantum */
    100,100,
    101,100,
    102,100,
    103,100,
    104,100,
    105,100,
    106,100,
    107,100,
    108,100,
    109,100,
    110,80,
    111,80,
    112,80,
    113,80,
    114,80,
    115,80,
    116,80,
    117,80,
    118,80,
    119,80,
    120,60,
    121,60,
    122,60,
    123,60,
    124,60,
    125,60,
    126,60,
    127,60,
    128,60,
    129,60,
    130,40,
    131,40,
    132,40,
    133,40,
    134,40,
EXAMPLE 2 Replacing The rt_dptbl Loadable Module  (Continued)

rt_dptbl(4)

rt_getdptbl()

return (config_rt_dptbl);
}

FILES

<sys/rt.h>

SEE ALSO

 priocntl(1), dispadmin(1M), priocntl(2), system(4)

System Administration Guide, Volume 1

System Interface Guide
sbus(4)

<table>
<thead>
<tr>
<th>NAME</th>
<th>sbus – configuration files for SBus device drivers</th>
</tr>
</thead>
</table>

**DESCRIPTION**

The SBus is a geographically addressed peripheral bus present on many SPARC hardware platforms. SBus devices are *self-identifying* — that is to say the SBus card itself provides information to the system so that it can identify the device driver that needs to be used. The device usually provides additional information to the system in the form of name-value pairs that can be retrieved using the DDI property interfaces. See `ddi_prop_op(9F)` for details.

The information is usually derived from a small Forth program stored in the FCode PROM on the card, so driver configuration files should be completely unnecessary for these devices. However, on some occasions, drivers for SBus devices may need to use driver configuration files to augment the information provided by the SBus card. See `driver.conf(4)` for further details.

When they are needed, configuration files for SBus device drivers should identify the parent bus driver implicitly using the `class` keyword. This removes the dependency on the particular bus driver involved since this may be named differently on different platforms.

All bus drivers of class `sbus` recognise the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>reg</strong></td>
<td>An arbitrary length array where each element of the array consists of a 3-tuple of integers. Each array element describes a logically contiguous mappable resource on the SBus. The first integer of each tuple specifies the slot number the card is plugged into. The second integer of each 3-tuple specifies the offset in the slot address space identified by the first element. The third integer of each 3-tuple specifies the size in bytes of the mappable resource. The driver can refer to the elements of this array by index, and construct kernel mappings to these addresses using <code>ddi_map_regs(9F)</code>. The index into the array is passed as the <code>rnumber</code> argument of <code>ddi_map_regs()</code>. You can use the <code>ddi_get*</code> and <code>ddi_put*</code> family of functions to access register space from a high-level interrupt context.</td>
</tr>
<tr>
<td><strong>interrupts</strong></td>
<td>An arbitrary length array where each element of the array consists of a single integer. Each array element describes a possible SBus interrupt level that the device might generate. The driver can refer to the elements of this array by index, and register interrupt handlers with the system using <code>ddi_add_intr(9F)</code>. The index into the array is passed as the <code>inumber</code> argument of <code>ddi_add_intr()</code>.</td>
</tr>
</tbody>
</table>
registers

An arbitrary length array where each element of the array consists of a 3-tuple of integers. Each array element describes a logically contiguous mappable resource on the SBus.

The first integer of each tuple should be set to $-1$, specifying that any SBus slot may be matched. The second integer of each 3-tuple specifies the offset in the slot address space identified by the first element. The third integer of each 3-tuple specifies the size in bytes of the mappable resource.

The registers property can only be used to augment an incompletely specified reg property with information from a driver configuration file. It may only be specified in a driver configuration file.

All SBus devices must provide reg properties to the system. The first two integer elements of the reg property are used to construct the address part of the device name under /devices.

Only devices that generate interrupts need to provide interrupts properties.

Occasionally, it may be necessary to override or augment the configuration information supplied by the SBus device. This can be achieved by writing a driver configuration file that describes a prototype device information (devinfo) node specification, containing the additional properties required.

For the system to merge the information, certain conditions must be met. First, the name property must be the same. Second, either the first two integers (slot number and offset) of the two reg properties must be the same, or the second integer (offset) of the reg and registers properties must be the same.

In the event that the SBus card has no reg property at all, the self-identifying information cannot be used, so all the details of the card must be specified in a driver configuration file.

**EXAMPLES**

**EXAMPLE 1** A sample configuration file.

Here is a configuration file for an SBus card called SUNW,netboard. The card already has a simple FCode PROM that creates name and reg properties, and will have a complete set of properties for normal use once the driver and firmware is complete.

In this example, we want to augment the properties given to us by the firmware. We use the same name property, and use the registers property to match the firmware reg property. That way we don’t have to worry about which slot the card is really plugged into.

We want to add an interrupts property while we are developing the firmware and driver so that we can start to experiment with interrupts. The device can generate interrupts at SBus level 3. Additionally, we want to set a debug-level property to 4.
EXAMPLE 1 A sample configuration file.

(Continued)

```
#  # Copyright (c) 1992, by Sun Microsystems, Inc.
#ident  "(#)SUNW,netboard.conf  1.4  92/03/10 SMI"
#
name="SUNW,netboard" class="sbus"
   registers=-1,0x40000,64,-1,0x80000,1024
   interrupts=3 debug-level=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>Architecture</td>
<td>SPARC</td>
</tr>
</tbody>
</table>

SEE ALSO  driver.conf(4), attributes(5), ddi_add_intr(9F), ddi_map_regs(9F),
           ddi_prop_op(9F)

Writing Device Drivers

WARNINGS  The wildcarding mechanism of the registers property matches every instance of
           the particular device attached to the system. This may not always be what is wanted.
NAME  sccsfile – format of an SCCS history file

DESCRIPTION An SCCS file is an ASCII file consisting of six logical parts:

- **checksum**: character count used for error detection
- **delta table**: log containing version info and statistics about each delta
- **usernames**: login names and/or group IDs of users who may add deltas
- **flags**: definitions of internal keywords
- **comments**: arbitrary descriptive information about the file
- **body**: the actual text lines intermixed with control lines

Each section is described in detail below.

Conventions Throughout an SCCS file there are lines which begin with the ASCII SOH (start of heading) character (octal 001). This character is hereafter referred to as the control character, and will be represented as ‘^A’. If a line described below is not depicted as beginning with the control character, it cannot do so and still be within SCCS file format.

Entries of the form `dddd` represent a five digit string (a number between 00000 and 99999).

Checksum The checksum is the first line of an SCCS file. The form of the line is:

```
^A h dddd
```

The value of the checksum is the sum of all characters, except those contained in the first line. The ^Ah provides a magic number of (octal) 064001.

Delta Table The delta table consists of a variable number of entries of the form:

```
^Aa inserted /deleted /unchanged

^Ad type sid yr /mo /da hr :mi :se username serial-number predecessor-sn

^Ai include-list

^Ax exclude-list

^Ag ignored-list

^Am mr-number

... comments ...

... comments ...

^Ae The first line (^Aa) contains the number of lines inserted/deleted/unchanged
respectively. The second line (^Aδ) contains the type of the delta (normal: D, and
removed: R), the SCCS ID of the delta, the date and time of creation of the delta, the
user-name corresponding to the real user ID at the time the delta was created, and the
serial numbers of the delta and its predecessor, respectively. The ^Ai, ^Ax, and ^Ag
lines contain the serial numbers of deltas included, excluded, and ignored,
respectively. These lines do not always appear.

The ^An lines (optional) each contain one MR number associated with the delta; the
^Ac lines contain comments associated with the delta.

The ^Ae line ends the delta table entry.

User Names
The list of user-names and/or numerical group IDs of users who may add deltas to
the file, separated by NEWLINE characters. The lines containing these login names
and/or numerical group IDs are surrounded by the bracketing lines ^Au and ^AU. An
empty list allows anyone to make a delta.

Flags
Flags are keywords that are used internally (see sccs-admin(1) for more information
on their use). Each flag line takes the form:

`^A^f flag optional text`
The following flags are defined in order of appearance:

`^A^f t type-of-program`
Defines the replacement for the 17:21:50 ID keyword.

`^A^f v program-name`
Controls prompting for MR numbers in addition to comments; if the optional text is
present it defines an MR number validity checking program.

`^A^f i`
Indicates that the 'No id keywords' message is to generate an error that
terminates the SCCS command. Otherwise, the message is treated as a warning
only.

`^A^f b`
Indicates that the -b option may be used with the SCCS get command to create a
branch in the delta tree.

`^A^f m module name`
Defines the first choice for the replacement text of the sccsfile.4 ID keyword.

`^A^f f floor`
Defines the “floor” release; the release below which no deltas may be added.

`^A^f c ceiling`
Defines the “ceiling” release; the release above which no deltas may be added.

`^A^f d default-sid`
The d flag defines the default SID to be used when none is specified on an SCCS
get command.
The n flag enables the SCCS delta command to insert a “null” delta (a delta that applies no changes) in those releases that are skipped when a delta is made in a new release (for example, when delta 5.1 is made after delta 2.7, releases 3 and 4 are skipped).

Enables the SCCS get command to allow concurrent edits of the same base SID.

 Defines a list of releases that are locked against editing.

 Defines the replacement for the ID keyword.

 The e flag indicates whether a source file is encoded or not. A 1 indicates that the file is encoded. Source files need to be encoded when they contain control characters, or when they do not end with a NEWLINE. The e flag allows files that contain binary data to be checked in.

**Comments**

Arbitrary text surrounded by the bracketing lines ^A and ^A. The comments section typically will contain a description of the file’s purpose.

**Body**

The body consists of text lines and control lines. Text lines do not begin with the control character, control lines do. There are three kinds of control lines: insert, delete, and end, represented by:

```
^AI ddddd
^AD ddddd
^AE ddddd
```

respectively. The digit string is the serial number corresponding to the delta for the control line.

**SEE ALSO**

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), sccs(1), what(1)
NAME  scsi – configuration files for SCSI target drivers

DESCRIPTION The architecture of the Solaris SCSI subsystem distinguishes two types of device
drivers: SCSI target drivers, and SCSI host adapter drivers. Target drivers like sd(7D)
and st(7D) manage the device on the other end of the SCSI bus. Host adapter drivers
manage the SCSI bus on behalf of all the devices that share it.

Drivers for host adapters provide a common set of interfaces for target drivers. These
interfaces comprise the Sun Common SCSI Architecture (SCSA) which are
documented as part of the Solaris DDI/DKI. See scsi_ifgetcap(9F),
scsi_init_pkt(9F), and scsi_transport(9F) for further details of these, and
associated routines.

Target drivers for SCSI devices should use a driver configuration file to enable them to
be recognized by the system.

Configuration files for SCSI target drivers should identify the host adapter driver
implicitly using the class keyword to remove any dependency on the particular host
adapter involved.

All host adapter drivers of class scsi recognize the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>target</td>
<td>Integer-valued SCSI target identifier that this driver will claim.</td>
</tr>
</tbody>
</table>
| lun      | Integer-valued SCSI logical unit number (LUN) that this driver
            will claim. |

All SCSI target drivers must provide target and lun properties. These properties are
used to construct the address part of the device name under /devices.

The SCSI target driver configuration files shipped with Solaris have entries for LUN 0
only. For devices that support other LUNs, such as some CD changers, the system
administrator may edit the driver configuration file to add entries for other LUNs.

EXAMPLES

EXAMPLE 1 A sample configuration file.

Here is a configuration file for a SCSI target driver called toaster.conf.

```
#   # Copyright (c) 1992, by Sun Microsystems, Inc.
#   #ident "@(#)toaster.conf 1.2 92/05/12 SMI"
name="toaster" class="scsi" target=4 lun=0;
```

Add the following lines to sd.conf for a six-CD changer on target 3, with LUNs 0 to

```
name="sd" class="scsi" target=3 lun=1;
name="sd" class="scsi" target=3 lun=2;
name="sd" class="scsi" target=3 lun=3;
name="sd" class="scsi" target=3 lun=4;
name="sd" class="scsi" target=3 lun=5;
```
EXAMPLE 1 A sample configuration file.  (Continued)

It is not necessary to add the line for LUN 0, as it already exists in the file shipped with Solaris.

SEE ALSO

driver.conf(4), sd(7D), st(7D), scsi_ifgetcap(9F), scsi_init_pkt(9F), scsi_transport(9F)

Writing Device Drivers

ANSI Small Computer System Interface-2 (SCSI-2)

NOTES

You need to ensure that the target and lun values claimed by your target driver do not conflict with existing target drivers on the system. For example, if the target is a direct access device, the standard sd.conf file will usually make sd claim it before any other driver has a chance to probe it.
**DESCRIPTION**

The `/var/yp/securenets` file defines the networks or hosts which are allowed access to information by the Network Information Service ("NIS").

The format of the file is as follows:

- Lines beginning with the "#" character are treated as comments.
- Otherwise, each line contains two fields separated by white space. The first field is a netmask, the second a network.
- The netmask field may be either `255.255.255.255` (IPv4), `ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff` (IPv6), or the string "host" indicating that the second field is a specific host to be allowed access.

Both `ypserv(1M)` and `ypxfrd(1M)` use the `/var/yp/securenets` file. The file is read when the `ypserv(1M)` and `ypxfrd(1M)` daemons begin. If `/var/yp/securenets` is present, `ypserv(1M)` and `ypxfrd(1M)` respond only to IP addresses in the range given. In order for a change in the `/var/yp/securenets` file to take effect, you must kill and restart any active daemons using `ypstop(1M)` and `ypstart(1M).

**EXAMPLES**

**EXAMPLE 1** Access for Individual Entries

If individual machines are to be given access, the entry could be:

```
255.255.255.255  192.9.1.20
```

or

```
host  192.0.1.20
```

**EXAMPLE 2** Access for a Class C Network

If access is to be given to an entire class C network, the entry could be:

```
255.255.255.0  192.9.1.0
```

**EXAMPLE 3** Access for a Class B Network

The entry for access to a class B network could be:

```
255.255.0.0  9.9.0.0
```

**EXAMPLE 4** Access for an Individual IPv6 Address

Similarly, to allow access for an individual IPv6 address:

```
ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff  fec0::111:abba:ace0:fba5e:1
```

or

```
host  fec0::111:abba:ace0:fba5e:1
```
EXAMPLE 4 Access for an Individual IPv6 Address  (Continued)

EXAMPLE 5 Access for all IPv6 Addresses Starting with fe80

To allow access for all IPv6 addresses starting with fe80:

```
ffff:: fe80::
```

FILES  
/var/yp/securenets  Configuration file for NIS security.

SEE ALSO  
ypserv(1M), ypstart(1M), ypstop(1M), ypxfrd(1M)

NOTES  
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. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.
services – Internet services and aliases

/etc/inet/services
/etc/services

The services file is a local source of information regarding each service available through the Internet. The services file can be used in conjunction with or instead of other services sources, including the NIS maps “services.byname” and the NIS+ table “services.” Programs use the getservbyname(3SOCKET) routines to access this information.

The services file contains an entry for each service. Each entry has the form:

```
service-name  port/protocol  aliases
```

- **service-name** This is the official Internet service name.
- **port / protocol** This field is composed of the port number and protocol through which the service is provided (for instance, 512/tcp).
- **aliases** This is a list of alternate names by which the service might be requested.

Fields can be separated by any number of SPACE and/or TAB characters. A ‘#’ (number sign) indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file.

Service names may contain any printable character other than a field delimiter, NEWLINE, or comment character.

/files

/etc/nsswitch.conf configuration file for name-service switch

SEE ALSO

getservbyname(3SOCKET), inetd.conf(4), nsswitch.conf(4)

NOTES

/etc/inet/services is the official SVR4 name of the services file. The symbolic link /etc/services exists for BSD compatibility.
shadow(4)

<table>
<thead>
<tr>
<th>NAME</th>
<th>shadow – shadow password file</th>
</tr>
</thead>
</table>
| DESCRIPTION | /etc/shadow is an access-restricted ASCII system file that stores users’ encrypted passwords and related information. The shadow file can be used in conjunction with other shadow sources, including the NIS maps passwd.byname and passwd.byuid and the NIS+ table passwd. Programs use the getspnam(3C) routines to access this information. 

The fields for each user entry are separated by colons. Each user is separated from the next by a newline. Unlike the /etc/passwd file, /etc/shadow does not have general read permission.

Each entry in the shadow file has the form:

```
```

The fields are defined as follows:

- **username**: The user’s login name (UID).
- **password**: A 13-character encrypted password for the user, a lock string to indicate that the login is not accessible, or no string, which shows that there is no password for the login.
- **lastchg**: The number of days between January 1, 1970, and the date that the password was last modified.
- **min**: The minimum number of days required between password changes.
- **max**: The maximum number of days the password is valid.
- **warn**: The number of days before password expires that the user is warned.
- **inactive**: The number of days of inactivity allowed for that user.
- **expire**: An absolute date specifying when the login may no longer be used.
- **flag**: Reserved for future use, set to zero. Currently not used.

The encrypted password consists of 13 characters chosen from a 64-character alphabet (., /, 0-9, A-Z, a-z). To update this file, use the passwd(1), useradd(1M), usermod(1M), or userdel(1M) commands.

In order to make system administration manageable, /etc/shadow entries should appear in exactly the same order as /etc/passwd entries; this includes “+” and “-” entries if the compat source is being used (see nsswitch.conf(4)).

<table>
<thead>
<tr>
<th>FILES</th>
<th>/etc/shadow shadow password file</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>/etc/passwd password file</td>
</tr>
</tbody>
</table>
If password aging is turned on in any name service the `passwd` line in the 
/etc/nsswitch.conf file must have a format specified in the nsswitch.conf(4) 
man page.

If the /etc/nsswitch.conf passwd policy is not in one of the supported formats, 
logins will not be allowed upon password expiration because the software does not 
know how to handle password updates under these conditions. See 
nsswitch.conf(4) for additional information.
sharetab(4)

NAME  sharetab – shared file system table

DESCRIPTION  sharetab resides in directory /etc/dfs and contains a table of local resources shared by the share command.

Each line of the file consists of the following fields:

pathname resource fstype specific_options description

where

pathname  Indicate the path name of the shared resource.
resource  Indicate the symbolic name by which remote systems can access the resource.
fstype  Indicate the file system type of the shared resource.
specific_options  Indicate file-system-type-specific options that were given to the share command when the resource was shared.
description  Describe the shared resource provided by the system administrator when the resource was shared.

SEE ALSO  share(1M)
### NAME
shells – shell database

### SYNOPSIS
/etc/shells

### DESCRIPTION
The shells file contains a list of the shells on the system. Applications use this file to determine whether a shell is valid. See getusershell(3C). For each shell a single line should be present, consisting of the shell’s path, relative to root.

A hash mark (#) indicates the beginning of a comment; subsequent characters up to the end of the line are not interpreted by the routines which search the file. Blank lines are also ignored.

The following default shells are used by utilities: /bin/bash, /bin/csh, /bin/jsh, /bin/ksh, /bin/pfcsh, /bin/pfksh, /bin/pfsh, /bin/sh, /bin/tcsh, /bin/zsh, /sbin/jsh, /sbin/sh, /usr/bin/bash, /usr/bin/csh, /usr/bin/jsh, /usr/bin/ksh, /usr/bin/pfcsh, /usr/bin/pfksh, /usr/bin/pfsh, and /usr/bin/sh, /usr/bin/tcsh, /usr/bin/zsh.

### FILES
/etc/shells lists shells on system

### SEE ALSO
vipw(1B), ftpr(1M), sendmail(1M), getusershell(3C), aliases(4)
NAME slp.conf – configuration file for Service Location Protocol agents

SYNOPSIS /etc/inet/slp.conf

DESCRIPTION slp.conf provides all Service Location Protocol ("SLP") agents with their operational configuration. slpd(1M) reads slp.conf on startup. Service Agents ("SAs") and User Agents ("UAs") read slp.conf on invocation of the SA and UA library routines; configuration parameters are then cached on a per-process basis. All SA's must use the same set of properties as slpd on the local machine, since slpd acts as an SA server.

The configuration file format consists of a newline-delimited list of zero or more property definitions. Each property definition corresponds to a particular configurable SLP, network, or other parameter in one or more of the three SLP agents. The file format grammar is shown in RFC 2234 as follows:

The properties fall into one of the following categories:

- DA Configuration
- Static Scope Configuration
- Tracing and Logging
- Serialized Proxy Registrations
- Networking Configuration Parameters
DA Configuration

The following are configuration properties and their parameters for DAs:

**net.slp.isDA**
- Setting Type: Boolean
- Default Value: False
- Range of Values: True or False
  A boolean that indicates whether slpd(1M) is to act as a DA. If False, slpd(1M) is not run as a DA.

**net.slp.DAHeartBeat**
- Setting Type: Integer
- Default Value: 10800 seconds (3 hours)
- Range of Values: 2000 – 259200000 seconds
  A 32-bit integer giving the number of seconds for the passive DA advertisement heartbeat. The default value is 10800 seconds. This property is ignored if net.slp.isDA is False.

**net.slp.DAAttributes**
- Setting Type: List of Strings
- Default Value: Unassigned
- Range of Values: List of Attribute Tag/Value List Pairs
  A comma-separated list of parenthesized attribute tag/value list pairs that the DA must advertise in DA advertisements. The property must be in the SLP attribute list wire format, which requires that you use a backslash ("\") to escape reserved characters. See RFC 2608 for more information on reserved characters, or refer to the Service Location Protocol Administration Guide.

Static Scope Configuration

The following properties and their parameters allow you to configure various aspects of scope and DA handling:

**net.slp.useScopes**
- Setting Type: List of Strings
- Default Value: Default, for SA and DA; unassigned for UA.
- Range of Values: List of Strings
  A list of strings indicating either the scopes that a UA or an SA is allowed to use when making requests, or the scopes a DA must support. If not present for the DA and SA, the default scope Default is used. If not present for the UA, then the user scoping model is in force, in which active and passive DA or SA discovery are used for scope discovery. The scope Default is used if no other information is available. If a DA or SA gets another scope in a request, a SCOPE_NOT_SUPPORTED error is returned, unless the request was multicast, in which case it is dropped. If a DA receives another scope in a registration, a SCOPE_NOT_SUPPORTED error will be returned. Unlike other
properties, this property is "read-only", so attempts to change it programatically after the configuration file has been read are ignored.

**net.slp.DAAddresses**

- **Setting Type**: List of Strings
- **Default Value**: Unassigned
- **Range of Values**: IPv4 addresses or host names

A list of IP addresses or DNS-resolvable names that denote the DAs to use for statically configured UAs and SAs. The property is read by slpd(1M), and registrations are forwarded to the DAs. The DAs are provided to UAs upon request. Unlike other properties, this property is "read-only", so attempts to change it after the configuration file has been read are ignored.

The following grammar describes the property:

```
addr-list   =  addr / addr "*" addr-list
addr        =  fqdn / hostnumber
fqdn        =  ALPHA / ALPHA *[ anum / "-" ] anum
anum        =  ALPHA / DIGIT
hostnumber  =  1*3DIGIT 3("." 1*3DIGIT)
```

The following is an example using this grammar:
```
sawah, mandi, sambal
```

IP addresses can be used instead of host names in networks where DNS is not deployed, but network administrators are reminded that using IP addresses will complicate machine renumbering, since the SLP configuration property files in statically configured networks will have to be changed.

**Tracing and Logging**

These properties direct tracing and logging information to be sent to syslogd at the LOG_INFO priority. These properties affect slpd(1M) only.

**net.slp.traceDATraffic**

- **Setting Type**: Boolean
- **Default Value**: False
- **Range of Values**: True or False

Set `net.slp.traceDATraffic` to `True` to enable logging of DA traffic by slpd.

**net.slp.traceMsg**

- **Setting Type**: Boolean
- **Default Value**: False
- **Range of Values**: True or False

Set `net.slp.traceMsg` to `True` to display details about SLP messages. The fields in all incoming messages and outgoing replies are printed by slpd.
### Serialized Proxy Registrations

#### net.slp.traceDrop
- **Setting Type**: Boolean
- **Default Value**: False
- **Range of Values**: True or False

Set this property to True to display details when an SLP message is dropped by slpd for any reason.

#### net.slp.traceReg
- **Setting Type**: Boolean
- **Default Value**: False
- **Range of Values**: True or False

Set this property to True to display the table of service advertisements when a registration or deregistration is processed by slpd.

### Networking Configuration Parameters

#### The following properties control reading and writing serialized registrations.

**net.slp.serializedRegURL**
- **Setting Type**: String
- **Default Value**: Unassigned
- **Range of Values**: Valid URL

A string containing a URL pointing to a document, which contains serialized registrations that should be processed when the slpd starts up.

#### The properties that follow allow you to set various network configuration parameters:

**net.slp.isBroadcastOnly**
- **Setting Type**: Boolean
- **Default Value**: False
- **Range of Values**: True or False

A boolean that indicates if broadcast should be used instead of multicast.

**net.slp.multicastTTL**
- **Setting Type**: Positive Integer
- **Default Value**: 255
- **Range of Values**: A positive integer from 1 to 255.

A positive integer less than or equal to 255 that defines the multicast TTL.
net.slp.DAActiveDiscoveryInterval

  Setting Type    Integer
  Default Value  900 seconds (15 minutes)
  Range of Values From 300 to 10800 seconds

  A 16-bit positive integer giving the number of seconds between DA active
  discovery queries. The default value is 900 seconds (15 minutes). If the property
  is set to zero, active discovery is turned off. This is useful when the DAs available are
  explicitly restricted to those obtained from the net.slp.DAAddresses property.

net.slp.multicastMaximumWait

  Setting Type    Integer
  Default Value  15000 milliseconds (15 seconds)
  Range of Values 1000 to 60000 milliseconds

  A 32-bit integer giving the maximum value for the sum of the
  net.slp.multicastTimeouts values and net.slp.DADiscoveryTimeouts
  values in milliseconds.

net.slp.multicastTimeouts

  Setting Type    List of Integers
  Default Value  3000,3000,3000,3000
  Range of Values List of Positive Integers

  A list of 32-bit integers used as timeouts, in milliseconds, to implement the
  multicast convergence algorithm. Each value specifies the time to wait before
  sending the next request, or until nothing new has been learned from two
  successive requests. In a fast network the aggressive values of
  1000,1250,1500,2000,4000 allow better performance. The sum of the list must
  equal net.slp.multicastMaximumWait.

net.slp.passiveDADetection

  Setting Type    Boolean
  Default Value  True
  Range of Values True or False

  A boolean indicating whether slpd should perform passive DA detection.

net.slp.DADiscoveryTimeouts

  Setting Type    List of Integers.
  Default Value  2000,2000,2000,2000,3000,4000
  Range of Values List of Positive Integers
A list of 32-bit integers used as timeouts, in milliseconds, to implement the multicast convergence algorithm during active DA discovery. Each value specifies the time to wait before sending the next request, or until nothing new has been learned from two successive requests. The sum of the list must equal `net.slp.multicastMaximumWait`.

**net.slp.datagramTimeouts**

<table>
<thead>
<tr>
<th>Setting Type</th>
<th>List of Integers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>3000, 3000, 3000</td>
</tr>
<tr>
<td>Range of Values</td>
<td>List of Positive Integers</td>
</tr>
</tbody>
</table>

A list of 32-bit integers used as timeouts, in milliseconds, to implement unicast datagram transmission to DAs. The nth value gives the time to block waiting for a reply on the nth try to contact the DA.

**net.slp.randomWaitBound**

<table>
<thead>
<tr>
<th>Setting Type</th>
<th>Integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>1000 milliseconds (1 second)</td>
</tr>
<tr>
<td>Range of Values</td>
<td>1000 to 3000 milliseconds</td>
</tr>
</tbody>
</table>

Sets the upper bound for calculating the random wait time before attempting to contact a DA.

**net.slp.MTU**

<table>
<thead>
<tr>
<th>Setting Type</th>
<th>Integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>1400</td>
</tr>
<tr>
<td>Range of Values</td>
<td>128 to 8192</td>
</tr>
</tbody>
</table>

A 16-bit integer that specifies the network packet size, in bytes. The packet size includes IP and TCP or UDP headers.

**net.slp.interfaces**

<table>
<thead>
<tr>
<th>Setting Type</th>
<th>List of Strings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>Default interface</td>
</tr>
<tr>
<td>Range of Values</td>
<td>IPv4 addresses or host names</td>
</tr>
</tbody>
</table>

List of strings giving the IP addresses or host names of the network interface cards on which the DA or SA should listen on port 427 for multicast, unicast UDP, and TCP messages. The default value is unassigned, indicating that the default network interface card should be used. An example is:

```
195.42.42.42, 195.42.142.1, 195.42.120.1
```

The example machine has three interfaces on which the DA should listen. Note that if IP addresses are used, the property must be renumbered if the network is renumbered.
The following configuration parameters apply to the UA:

**net.slp.locale**
- **Setting Type**: String
- **Default Value**: en
- **Range of Values**: See RFC 1766 for a list of the locale language tag names.

A RFC 1766 Language Tag for the language locale. Setting this property causes the property value to become the default locale for SLP messages.

**net.slp.maxResults**
- **Setting Type**: Integer
- **Default Value**: -1
- **Range of Values**: -1, positive integer

A 32 bit-integer that specifies the maximum number of results to accumulate and return for a synchronous request before the timeout, or the maximum number of results to return through a callback if the request results are reported asynchronously. Positive integers and -1 are legal values. If the value of `net.slp.maxResults` is -1, all results should be returned.

**net.slp.typeHint**
- **Setting Type**: List of Strings
- **Default Value**: Unassigned
- **Range of Values**: Service type names

A list of service type names. In the absence of any DAs, UAs perform SA discovery to find scopes. If the `net.slp.typeHint` property is set, only SA's advertising types on the list respond. Note that UAs set this property programmatically. It is not typically set in the configuration file. The default is unassigned, meaning do not restrict the type.

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWslnpr</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Standard</td>
</tr>
</tbody>
</table>

**SEE ALSO**

slpd(1M), slpd.reg(4), slp_api(3SLP), slp(7P)
Service Location Protocol Administration Guide


NAME
sldp.reg – serialized registration file for the service location protocol daemon (sldp)

SYNOPSIS
/etc/inet/sldp.reg

DESCRIPTION
The serialized registration file contains a group of registrations that sldpd(1M) registers when it starts. These registrations are primarily for older service programs that do not internally support SLP and cannot be converted. The character format of the registration file is required to be ASCII. To use serialized registrations, set the net.slp.serializedRegURL property in sldp.conf(4) to point at a valid sldp.reg file. The syntax of the serialized registration file, in ABNF format (see RFC 2234), is as follows:

```
ser-file  =  reg-list
reg-list  =  reg / reg reg-list
reg       =  creg / ser-reg
creg      =  comment-line ser-reg
comment-line  =  ( "#" / ";" ) *allchar newline
ser-reg   =  url-props [slist] [attr-list] newline
url-props =  surl *," lang ",", ltime [ "," type ] newline
surl      = ;The registration’s URL. See
            ;[8] for syntax.
lang      =  *8ALPHA [ *-* *8ALPHA ]
            ;RFC 1766 Language Tag see [6].
ltime     =  *5DIGIT
            ; A positive 16-bit integer
            ; giving the lifetime
            ; of the registration.
type      =  ; The service type name, see [7]
            ; and [8] for syntax.
slist     =  "scopes" =" scope-list newline
scope-list =  scope-name / scope-name ," scope-list
scope     =  ; See grammar of [7] for
            ; scope-name syntax.
attr-list  =  attr-def / attr-def attr-list
attr-def   =  ( attr / keyword ) newline
keyword    =  attr-id
attr       =  attr-id =" attr-val-list
attr-id    =  ;Attribute id, see [7] for syntax.
attr-val-list =  attr-val / attr-val *," attr-val-list
attr-val   =  ;Attribute value, see [7] for syntax
allchar    =  char / WSP
char       =  *8DIGIT / *8ALPHA / other
other      =  %x21-%x2f / %x3a-%x40 /
            %x5b-%x60 / %7b-%7e
            ; All printable, nonwhitespace US-ASCII
            ; characters.
newline    =  C R / ( CRLF )
```

The syntax for attributes and attribute values requires that you use a backslash to escape special characters, in addition to non-ASCII characters, as specified in RFC 2608. The sldpd command handles serialized registrations exactly as if they were registered by an SA. In the url-props production, the type token is optional. If the type token is present for a service: URL, a warning is signalled, and the type name is ignored. If the maximum lifetime of 65535 seconds is specified, the registration is taken to be permanent, and it is continually refreshed by the DA or SA server until it exits.
Scopes can be included in a registration by including an attribute definition with tag `scopes` followed by a comma-separated list of scope names immediately after the `url-prop` production. If the optional `scope-list` is present, the registrations are made in the indicated scopes; otherwise, they are registered in the scopes with which the DA or SA server was configured through the `net.slp.useScopes` property. If any conflicts occur between the scope list and the `net.slp.useScopes` property, an error message is issued by way of `syslog(3C)`. Refer to information regarding `LOG_INFO` in `syslog(3C)`.

Service advertisements are separated by a single blank line. Additionally, the file must end with a single blank line.

**EXAMPLE 1** Using a Serialized Registration File

The following serialized registration file shows an instance of the service type `foo`, with a lifetime of 65535 seconds, in the `en` locale, with scope `somescope`:

```plaintext
# register foo
service:foo://fooserver/foopath,en,65535
scopes=somescope
description=bogus
security=kerberos_v5
location=headquarters

# next registration...
```

**ATTRIBUTES**

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

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>SUNWslpr</td>
</tr>
<tr>
<td>CSI</td>
<td>Enabled</td>
</tr>
<tr>
<td>Interface Stability</td>
<td>Standard</td>
</tr>
</tbody>
</table>

**SEE ALSO**

`sldpd(1M), sld_api(3SLP), syslog(3C), sldp.conf(4), attributes(5)`


NAME  sock2path – file that maps sockets to transport providers

SYNOPSIS  /etc/sock2path

DESCRIPTION  The socket mapping file, /etc/sock2path, is a system file that contains the mappings between the socket(3SOCKET) call parameters and the transport provider driver. Its format is described on the soconfig(1M) manual page.

The init(1M) utility uses the soconfig utility with the sock2path file during the booting sequence.

EXAMPLES

EXAMPLE 1 A Sample sock2path File

The following is a sample sock2path file:

```
# Family Type Protocol Path
 2  2  0  /dev/tcp
 2  2  6  /dev/tcp
 26 2  0  /dev/tcp6
 26 2  6  /dev/tcp6
 2  1  0  /dev/udp
 2  1  17 /dev/udp
 26 1  0  /dev/udp6
 26 1  17 /dev/udp6
 1  2  0  /dev/ticotsord
 1  6  0  /dev/ticotsord
 1  1  0  /dev/ticots
 2  4  0  /dev/rawip
 26 4  0  /dev/rawip6
 24 4  0  /dev/rts
 27 4  2  /dev/keysock
```

SEE ALSO  soconfig(1M), socket(3SOCKET)

Network Interface Guide
NAME  space – disk space requirement file

DESCRIPTION  space is an ASCII file that gives information about disk space requirements for the target environment. The space file defines space needed beyond what is used by objects defined in the prototype(4) file; for example, files which will be installed with the installf(1M) command. The space file should define the maximum amount of additional space that a package will require.

The generic format of a line in this file is:

pathname blocks inodes

Definitions for the fields are as follows:

pathname  Specify a directory name which may or may not be the mount point for a filesystem. Names that do not begin with a slash (’/’) indicate relocatable directories.

blocks  Define the number of disk blocks required for installation of the files and directory entries contained in the pathname (using a 512-byte block size).

inodes  Define the number of inodes required for installation of the files and directory entries contained in the pathname.

EXAMPLES  EXAMPLE 1  A sample file.

# extra space required by config data which is
dynamically loaded onto the system

data  500  1

SEE ALSO  installf(1M), prototype(4)

Application Packaging Developer’s Guide
The `sulog` file is a record of all attempts by users on the system to execute the `su(1M)` command. Each time `su(1M)` is executed, an entry is added to the `sulog` file.

Each entry in the `sulog` file is a single line of the form:

```
SU date time result port user newuser
```

where

- **date**: The month and date `su(1M)` was executed. `date` is displayed in the form `mm/dd` where `mm` is the month number and `dd` is the day number in the month.
- **time**: The time `su(1M)` was executed. `time` is displayed in the form `HH/MM` where `HH` is the hour number (24 hour system) and `MM` is the minute number.
- **result**: The result of the `su(1M)` command. A `+` sign is displayed in this field if the `su` attempt was successful; otherwise a `-` sign is displayed.
- **port**: The name of the terminal device from which `su(1M)` was executed.
- **user**: The user id of the user executing the `su(1M)` command.
- **newuser**: The user id being switched to with `su(1M)`.

**EXAMPLES**

**EXAMPLE 1** A sample `sulog` file.

Here is a sample `sulog` file:

```
SU 02/25 09:29 + console root-sys
SU 02/25 09:32 + pts/3 user1-root
SU 03/02 08:03 + pts/5 user1-root
SU 03/03 08:19 + pts/5 user1-root
SU 03/09 14:24 - pts/5 guest3-root
SU 03/09 14:24 - pts/5 guest3-root
SU 03/14 08:31 + pts/4 user1-root
```

**FILES**

```
/var/adm/sulog su log file
/etc/default/su contains the default location of sulog
```

**SEE ALSO**

`su(1M)`
Solaris (Intel Platform Edition) supports the ISA and EISA buses as the system bus. Drivers for devices on these buses use the device tree built by the booting system to retrieve the necessary system resources used by the driver. These resources include device I/O port addresses, any interrupt capabilities that the device may have, any DMA channels it may require, and any memory-mapped addresses it may occupy.

Configuration files for ISA and EISA device drivers are only necessary to describe properties used by a particular driver that are not part of the standard properties found in the device tree. See `driver.conf(4)` for further details of configuration file syntax.

The ISA and EISA nexus drivers all belong to class `sysbus`. All bus drivers of class `sysbus` recognize the following properties:

**interrupts**

An arbitrary-length array where each element of the array represents a hardware interrupt (IRQ) that is used by the device. In general, this array only has one entry unless a particular device uses more than one IRQ.

Solaris defaults all ISA and EISA interrupts to IPL 5. This interrupt priority may be overridden by placing an `interrupt-priorities` property in a .conf file for the driver. Each entry in the array of integers for the `interrupt-priorities` property is matched one-to-one with the elements in the `interrupts` property to specify the IPL value that will be used by the system for this interrupt in this driver. This is the priority that this device’s interrupt handler will receive relative to the interrupt handlers of other drivers. The priority is an integer from 1 to 16. Generally, disks are assigned a priority of 5, while mice and printers are lower, and serial communication devices are higher, typically 7. 10 is reserved by the system and must not be used. Priorities 11 and greater are high level priorities and are generally not recommended (see `ddi_intr_hilevel(9F)`).

The driver can refer to the elements of this array by index using `ddi_add_intr(9F)`. The index into the array is passed as the `inumber` argument of `ddi_add_intr()`.

Only devices that generate interrupts will have an `interrupts` property.

**reg**

An arbitrary-length array where each element of the array consists of a 3-tuple of integers. Each array element describes a contiguous memory address range associated with the device on the bus.
The first integer of the tuple specifies the memory type, 0 specifies a memory range and 1 specifies an I/O range. The second integer specifies the base address of the memory range. The third integer of each 3-tuple specifies the size, in bytes, of the mappable region.

The driver can refer to the elements of this array by index, and construct kernel mappings to these addresses using ddi_map_regs(9F). The index into the array is passed as the rnumber argument of ddi_map_regs().

All sysbus devices will have reg properties. The first tuple of this property is used to construct the address part of the device name under /devices. In the case of Plug and Play ISA devices, the first tuple is a special tuple that does not denote a memory range, but is used by the system only to create the address part of the device name. This special tuple can be recognized by determining if the top bit of the first integer is set to a one.

The order of the tuples in the reg property is determined by the boot system probe code and depends on the characteristics of each particular device. However, the reg property will maintain the same order of entries from system boot to system boot. The recommended way to determine the reg property for a particular device is to use the prtconf(1M) command after installing the particular device. The output of the prtconf command can be examined to determine the reg property for any installed device.

You can use the ddi_get* and ddi_put* family of functions to access register space from a high-level interrupt context.

dma-channels A list of integers that specifies the DMA channels used by this device. Only devices that use DMA channels will have a dma-channels property.

It is recommended that drivers for devices connected to the system bus recognize the following standard property names:

slot The number of the slot containing the device, if known. (Only for EISA 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>Architecture</td>
<td>IA</td>
</tr>
</tbody>
</table>
Writing Device Drivers

SEE ALSO

prtconf(1M), driver.conf(4), scsi(4), attributes(5), ddi_add_intr(9F),
ddi_intr_hilevel(9F), ddi_map_regs(9F), ddi_prop_op(9F)

Writing Device Drivers
sysidcfg(4)

NAME | sysidcfg – system identification configuration file

DESCRIPTION

When a diskless client boots for the first time or a system installs over the network, the booting software tries to obtain configuration information about the system (such as the system’s root password or name service) from, first, a sysidcfg file and then the name service databases. If the booting software cannot find the information, it prompts the user for it. Like the name service databases, the sysidcfg file can be used to avoid the user prompts and provide a totally hands-off booting process.

The sysidcfg file preconfigures information through a set of keywords. You can specify one or more of the keywords to preconfigure as much information as you want. Each set of systems (one or more) that has unique configuration information must have its own sysidcfg file. For example, you can use the same sysidcfg file to preconfigure the time zone for multiple systems if you want all the systems to have the same time zone configured. However, if you want to preconfigure a different root password for each of those systems, then each system would need its own sysidcfg file.

Where To Put the sysidcfg File

The sysidcfg file can reside on a shared NFS network directory or the root directory on a UFS or PCFS diskette in the system’s diskette drive. If you put the sysidcfg file on a shared NFS network directory, you have to use the -p option of the add_install_client(1M) command (see install_scripts(1M)) to specify where the system being installed can find the sysidcfg file. If you put the sysidcfg file on a diskette, you need to make sure the diskette is in the system’s diskette drive when the system boots (on IA systems, the sysidcfg file should reside on the Solaris Device Configuration Assistant diskette).

Only one sysidcfg file can reside in a directory or diskette. If you are creating more than one sysidcfg file, they must reside in different directories or diskettes.

Keyword Syntax Rules

The following rules apply to the keywords in a sysidcfg file:

- Keywords can be in any order
- Keywords are not case-sensitive
- Keyword values can be optionally enclosed in single (’) or double (“) quotes
- Only the first instance of a keyword is valid; if you specify the same keyword more than once, the first keyword specified will be used.

Keywords – All Platforms

The following keywords apply to both SPARC and IA platforms.

Name Service, Domain Name, Name Server

Naming-related keywords are as follows:

name_service=NIS,NIS+,LDAP,DNS,NONE

domain_name=domain_name
name_server=hostname (ip_address)

For the NIS and NIS+ keywords, the options are:
The following is an example NIS entry:

```plaintext
ame_service=NIS
{domain_name=west.arp.com name_server=timber(129.221.2.1)}
```

For NIS+, the example is identical to the one above, except for the replacement of the keyword NIS by NIS+.

For DNS, the syntax is:

```plaintext
domain_name=domain_name; name_server=ip_address, ... ;
search=domain_name, ...
```

You can have a maximum of three IP addresses and six domain names. The total length of a search entry cannot exceed 250 characters. The following is an example DNS entry:

```plaintext
name_service=DNS
{domain_name=west.arp.com
name_server=10.0.1.10,10.0.1.20
search=arp.com,east.arp.com}
```

For LDAP, the syntax is:

```plaintext
domain_name=domain_name;
profile=profile_name;
profile_server=ip_address
```

The following is an example LDAP entry:

```plaintext
name_service=LDAP
{domain_name=west.arp.com
profile=default
profile_server=129.221.2.1}
```

Choose only one value for name_service. Include either, both, or neither of the domain_name and name_server keywords, as needed. If no keywords are used, omit the curly braces.

**Network Interface, Hostname, IP address, Netmask, DHCP, IPv6, Default Route**

Network-related keywords are as follows:

```plaintext
network_interface=NONE, PRIMARY, value
```

If you are using DHCP, the options for PRIMARY and value are:

```plaintext
dhcp; protocol_ipv6=yes_or_no
```

For example:

```plaintext
network_interface=primary {dhcp protocol_ipv6=yes}
```

If you are not using DHCP, the options for PRIMARY and value are:

```plaintext
hostname=host_name;
ip_address=ip_address;
etmask=netmask;
```
protocol_ipv6=yes_or_no
default_route=ip_address (IPv4 address only)

For example:

```
network_interface=le0
{hostname=feron
 ip_address=129.222.2.1
 netmask=255.255.0.0
 protocol_ipv6=no
 default_route=129.222.2.1}
```

Choose only one value for network_interface. Include any combination or none of
the hostname, ip_address, netmask, and default_route keywords, as needed.
If you do not use any of these keywords, omit the curly braces.

protocol_ipv6 and default_route are optional; you do not need to specify them.
default_route accepts only an IPv4 address.

**Root Password**

The root password keyword is root_password. Possible values are encrypted from
/etc/shadow. Syntax is:

```
root_password=encrypted_password
```

**Security Policy**

The security—related keyword is security_policy. It has the following syntax:

```
security_policy=kerberos, NONE
```

The kerberos keyword has the following options:

```
{default_realm=FQDN admin_server=FQDN kdc=FQDN1, FQDN2, FQDN3}
```

where FQDN is a fully qualified domain name. An example of the security_policy
keyword is as follows:

```
security_policy=kerberos {default_realm=YourSite.COM
 admin_server=krbadmin.YourSite.COM
 kdc=kdc1.YourSite.COM, kdc2.YourSite.COM}
```

You can list a maximum of three key distribution centers (KDCs) for a
security_policy keyword. At least one is required.

**Language in Which to Display the Install Program**

The system-location keyword is system_locale. It has the following syntax:

```
system_locale=locale
```

where locale is /usr/lib/locale.

**Terminal Type**

The terminal keyword is terminal. It has the following syntax:
terminal=terminal_type

where *terminal_type* is a value from /usr/share/lib/terminfo/*.

**Timezone Information**

The timezone keyword is timezone. It has the following syntax:

timezone=timezone

where *timezone* is a value from /usr/share/lib/zoneinfo/*.

**Date and Time**

The time server keyword is timeserver. It has the following syntax:

timeserver=localhost
timeserver=hostname
timeserver=ip_address

If you specify localhost as the time server, the system’s time is assumed to be correct. If you specify the hostname or *ip_address* (if you are not running a name service) of a system, that system’s time is used to set the time.

**Keywords — IA Platform**

The following keywords apply only to IA platforms. For all these keywords, use kdmconfig -d to create or append to the sysidcfg file. See kdmconfig(1M)

Monitor type

The monitor—related keyword is monitor. The syntax is:

monitor=monitor_type

Keyboard language, keyboard layout

The keyboard—language keyword is keyboard. The syntax is:

keyboard=keyboard_language {layout=value}

Graphics card, color depth, display resolution, screen size

The display-related keywords are display, size, depth, and resolution. The syntax is:

display=graphics_card {size=screen_size depth=color_depth resolution=screen_resolution}

Pointing device, number of buttons, IRQ level

The mouse-related keywords are pointer, nbuttons, and irq.

pointer=pointing_device {nbuttons=number_buttons irq=value}

**EXAMPLE 1** Sample sysidcfg files

The following example is a sysidcfg file for a group of SPARC systems to install over the network. (The host names, IP addresses, and netmask of these systems have been preconfigured by editing the name service.) Because all the system configuration information has been preconfigured, an automated installation can be created by using a custom JumpStart profile.
EXAMPLE 1 Sample \texttt{sysidcfg} files (Continued)

\begin{verbatim}
   system_locale=en_US
   timezone=US/Central
timeserver=localhost
terminal=sun-cmd
name_service=NIS {domain_name=marquee.central.sun.com
   name_server=connor(129.152.112.3)}
root_password=m4OPOWNY
security_policy=kerberos
   {default_realm=Yoursite.COM
    admin_server=krbadmin.Yoursite.COM
    kdc=kdc1.Yoursite.COM, kdc2.Yoursite.COM}
\end{verbatim}

The following example is a \texttt{sysidcfg} file created for a group of IA systems to install over the network that all have the same keyboard, graphics cards, and pointing devices. The device information (keyboard, display, and pointer) was captured from running \texttt{kdmconfig-d} (see \texttt{kdmconfig(1M)}). In this example, users would see only the prompt to select a language (\texttt{system Locale}) for displaying the rest of the Solaris installation program.

\begin{verbatim}
   keyboard=ATKBBD {layout=US-English}
display=ati {size=15-inch}
pointer=MS-S
timezone=US/Central
timeserver=connor
terminal=AT386
name_service=NIS {domain_name=marquee.central.sun.com
   name_server=connor(129.152.112.3)}
root_password=URFUni9
security_policy=none
\end{verbatim}

SEE ALSO \texttt{install_scripts(1M)}, \texttt{kdmconfig(1M)}, \texttt{sysidtool(1M)}

\textit{Solaris 8 Advanced Installation Guide}
The file /etc/syslog.conf contains information used by the system log daemon, syslogd(1M), to forward a system message to appropriate log files and/or users. syslogd preprocesses this file through m4(1) to obtain the correct information for certain log files, defining LOGHOST if the address of "loghost" is the same as one of the addresses of the host that is running syslogd.

A configuration entry is composed of two TAB-separated fields:

```
selector  action
```

The `selector` field contains a semicolon-separated list of priority specifications of the form:

```
facility.level  ;  facility.level
```

where `facility` is a system facility, or comma-separated list of facilities, and `level` is an indication of the severity of the condition being logged. Recognized values for `facility` include:

- `user` Messages generated by user processes. This is the default priority for messages from programs or facilities not listed in this file.
- `kern` Messages generated by the kernel.
- `mail` The mail system.
- `daemon` System daemons, such as in.ftpd(1M)
- `auth` The authorization system: login(1), su(1M), getty(1M), among others.
- `lpr` The line printer spooling system: lpr(1B), lpc(1B), among others.
- `news` Reserved for the USENET network news system.
- `uucp` Reserved for the UUCP system; it does not currently use the syslog mechanism.
- `cron` The cron/at facility; crontab(1), at(1), cron(1M), among others.
- `local0-7` Reserved for local use.
- `mark` For timestamp messages produced internally by syslogd.
- `*` An asterisk indicates all facilities except for the `mark` facility.

Recognized values for `level` are (in descending order of severity):

- `emerg` For panic conditions that would normally be broadcast to all users.
<table>
<thead>
<tr>
<th>Facility</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alert</td>
<td>For conditions that should be corrected immediately, such as a corrupted system database.</td>
</tr>
<tr>
<td>crit</td>
<td>For warnings about critical conditions, such as hard device errors.</td>
</tr>
<tr>
<td>err</td>
<td>For other errors.</td>
</tr>
<tr>
<td>warning</td>
<td>For warning messages.</td>
</tr>
<tr>
<td>notice</td>
<td>For conditions that are not error conditions, but may require special handling. A configuration entry with a level value of notice must appear on a separate line.</td>
</tr>
<tr>
<td>info</td>
<td>Informational messages.</td>
</tr>
<tr>
<td>debug</td>
<td>For messages that are normally used only when debugging a program.</td>
</tr>
</tbody>
</table>
| none     | Do not send messages from the indicated facility to the selected file. For example, a selector of

\[*.debug;mail.none\]

will send all messages except mail messages to the selected file. |

The action field indicates where to forward the message. Values for this field can have one of four forms:

- A filename, beginning with a leading slash, which indicates that messages specified by the selector are to be written to the specified file. The file will be opened in append mode.
- The name of a remote host, prefixed with an @, as with: @server, which indicates that messages specified by the selector are to be forwarded to the syslogd on the named host. The hostname "loghost" is the hostname given to the machine that will log syslogd messages. Every machine is "loghost" by default. See /etc/hosts. It is also possible to specify one machine on a network to be "loghost" by making the appropriate host table entries. If the local machine is designated to be "loghost", then syslogd messages are written to the appropriate files. Otherwise, they are sent to the machine "loghost" on the network.
- A comma-separated list of usernames, which indicates that messages specified by the selector are to be written to the named users if they are logged in.
- An asterisk, which indicates that messages specified by the selector are to be written to all logged-in users.

Blank lines are ignored. Lines for which the first nonwhite character is a ‘#’ are treated as comments.
EXAMPLE 1 A Sample Configuration File

With the following configuration file:

```
*.notice /var/log/notice
mail.info /var/log/notice
*.crit /var/log/critical
kern,mark.debug /dev/console
kern.err @server
*.emerg *
*.alert root,operator
*.alert;auth.warning /var/log/auth
```

**syslogd**(1M) will log all mail system messages except debug messages and all notice (or higher) messages into a file named `/var/log/notice`. It logs all critical messages into `/var/log/critical`, and all kernel messages and 20-minute marks onto the system console.

Kernel messages of err (error) severity or higher are forwarded to the machine named server. Emergency messages are forwarded to all users. The users root and operator are informed of any alert messages. All messages from the authorization system of warning level or higher are logged in the file `/var/log/auth`.

**FILES**

```
/var/log/notice
```

log of all mail system messages (except debug messages) and all messages of notice level or higher.

```
/var/log/critical
```

log of all critical messages

```
/var/log/auth
```

log of all messages from the authorization system of warning level or higher

**SEE ALSO**
at(1), crontab(1), logger(1), login(1), lp(1), lpc(1B), lpr(1B), m4(1), cron(1M), getty(1M), in.ftpd(1M), su(1M), syslogd(1M), syslog(3C), hosts(4)
<table>
<thead>
<tr>
<th>NAME</th>
<th>system – system configuration information file</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The <code>system</code> file is used for customizing the operation of the operating system kernel. The recommended procedure is to preserve the original <code>system</code> file before modifying it.</td>
</tr>
<tr>
<td></td>
<td>The <code>system</code> file contains commands which are read by the kernel during initialization and used to customize the operation of your system. These commands are useful for modifying the system’s treatment of its loadable kernel modules.</td>
</tr>
<tr>
<td></td>
<td>The syntax of the <code>system</code> file consists of a list of keyword/value pairs which are recognized by the system as valid commands. Comment lines must begin with an asterisk (<code>*</code>) and end with a newline character. All commands are case-insensitive except where noted. A command line can be no more than 80 characters in length.</td>
</tr>
<tr>
<td></td>
<td>Commands that modify the system’s operation with respect to loadable kernel modules require you to specify the module type by listing the module’s namespace. The following namespaces are currently supported:</td>
</tr>
<tr>
<td></td>
<td><strong>drv</strong> Modules in this namespace are device drivers.</td>
</tr>
<tr>
<td></td>
<td><strong>exec</strong> Modules in this namespace are execution format modules. The following <code>exec</code> modules are currently provided:</td>
</tr>
<tr>
<td></td>
<td>Only on SPARC system:</td>
</tr>
<tr>
<td></td>
<td><code>aoutexec</code></td>
</tr>
<tr>
<td></td>
<td>Only on IA system:</td>
</tr>
<tr>
<td></td>
<td><code>coffexec</code></td>
</tr>
<tr>
<td></td>
<td>On SPARC and IA systems:</td>
</tr>
<tr>
<td></td>
<td><code>elfexec</code></td>
</tr>
<tr>
<td></td>
<td><code>intpexec</code></td>
</tr>
<tr>
<td></td>
<td><code>javaexec</code></td>
</tr>
<tr>
<td></td>
<td><strong>fs</strong> These modules are filesystems.</td>
</tr>
<tr>
<td></td>
<td><strong>sched</strong> These modules implement a process scheduling algorithm.</td>
</tr>
<tr>
<td></td>
<td><strong>strmod</strong> These modules are STREAMS modules.</td>
</tr>
<tr>
<td></td>
<td><strong>sys</strong> These modules implement loadable system-call modules.</td>
</tr>
<tr>
<td></td>
<td><strong>misc</strong> These modules do not fit into any of the above categories, so are considered &quot;miscellaneous&quot; modules.</td>
</tr>
</tbody>
</table>
Below is a description of each of the supported commands:

**exclude:**

<namespace>/<modulename>

Do not allow the listed loadable kernel module to be loaded. `exclude` commands are cumulative; the list of modules to exclude is created by combining every `exclude` entry in the system file.

**include:**

<namespace>/<modulename>

Include the listed loadable kernel module. This is the system’s default, so using `include` does not modify the system’s operation. `include` commands are cumulative.

**forceload:**

<namespace>/<modulename>

Force this kernel module to be loaded during kernel initialization. The default action is to automatically load the kernel module when its services are first accessed. `forceload` commands are cumulative.

**rootdev:** <device name>

Set the root device to the listed value instead of using the default root device as supplied by the boot program.

**rootfs:** <root filesystem type>

Set the root filesystem type to the listed value.

**moddir:** <first module path>[[; ]<second ...>][...]

Set the search path for loadable kernel modules. This command operates very much like the `PATH` shell variable. Multiple directories to search can be listed together, delimited either by blank spaces or colons.

**set** [<module>]:<symbol> [=, |, & ]<value>

Set an integer or character pointer in the kernel or in the selected kernel module to a new value. This command is used to change kernel and module parameters and thus modify the operation of your system. Assignment operations are not cumulative, whereas bitwise AND and OR operations are cumulative.

Operations that are supported for modifying integer variables are: simple assignment, inclusive bitwise OR, bitwise AND, one’s complement, and negation. Variables in a specific loadable module can be targeted for modification by specifying the variable name prefixed with the kernel module name and a colon (:) separator. Values can be specified as hexadecimal (0x10), Octal (046), or Decimal (5).

The only operation supported for modifying character pointers is simple assignment. Static string data such as character arrays cannot be modified using the `set` command. Use care and ensure that the variable you are modifying is in fact a character pointer. The `set`
command is very powerful, and will likely cause problems if used carelessly. The entire command, including the quoted string, cannot exceed 80 characters. The following escape sequences are supported within the quoted string:

\n (newline)
\t (tab)
\b (backspace)

### EXAMPLE 1
A sample system file.

The following is a sample system file.

- Force the ELF exec kernel module to be loaded during kernel initialization. Execution type modules are in the exec namespace.
  - forceload: exec/elfexec
- Change the root device to /sbus@1,f8000000/esp@0,800000/sd@3,0:a.
- You can derive root device names from /devices.
- Root device names must be the fully expanded Open Boot Prom device name. This command is platform and configuration specific.
- This example uses the first partition (a) of the SCSI disk at SCSI target 3 on the esp host adapter in slot 0 (on board) of the SBus of the machine.
- Adapter unit-address 3,0 at sbus unit-address 0,800000.
  - rootdev: /sbus@1,f8000000/esp@0,800000/sd@3,0:a
- Set the filesystem type of the root to ufs. Note that the equal sign can be used instead of the colon.
  - rootfs: ufs
- Set the search path for kernel modules to look first in /usr/phil/mod_test for modules, then in /kernel/modules (the default) if not found. Useful for testing new modules.
  - Note that you can delimit your module pathnames using colons instead of spaces: moddir:/newmodules:/kernel/modules
  - moddir:/usr/phil/mod_test /kernel/modules.
- Set the configuration option `_POSIX_CHOWN_RESTRICTED`:
  - This configuration option is enabled by default.
  - `set rstchown = 1`
  - `set rstchown = 0`
- Set the integer variable "maxusers" in the kernel to 16. This is a useful tuning parameter.
  - `set maxusers = 16`
- Turn on debugging messages in the modules mydriver. This is useful during driver development.
  - `set mydriver:debug = 1`
- Bitwise AND the kernel variable "moddebug" with the one’s complement of the hex value 0x880, and set "moddebug" to this new value.
  - `set moddebug & ~0x880`
- Demonstrate the cumulative effect of the SET bitwise AND/OR operations by further modifying "moddebug" by ORing it with 0x40.
  - `set moddebug | 0x40`
EXAMPLE 1 A sample system file.  (Continued)

WARNINGS  system file lines must be fewer than 80 characters in length.

Use care when modifying the system file; it modifies the operation of the kernel. If you preserved the original system file, you can boot using boot -a, which will ask you to specify the path to the saved file. This should allow the system to boot correctly. If you cannot locate a system file that will work, you may specify /dev/null. This acts as an empty system file, and the system will attempt to boot using its default settings.

NOTES  /etc/system is only read once; at boot time.
### NAME
telnetrc – file for telnet default options

### DESCRIPTION
The `.telnetrc` file contains commands that are executed when a connection is established on a per-host basis. Each line in the file contains a host name, one or more spaces or tabs, and a `telnet(1)` command. The host name, DEFAULT, matches all hosts. Lines beginning with the pound sign (#) are interpreted as comments and therefore ignored. `telnet(1)` commands are case-insensitive to the contents of the `.telnetrc` file.

The `.telnetrc` file is retrieved from each user’s HOME directory.

### EXAMPLES
#### EXAMPLE 1 A sample file.

In the following example, a `.telnetrc` file executes the `telnet(1)` command, `toggle`:

```bash
weirdhost toggle crmod
# Always export $PRINTER
DEFAULT environ export PRINTER
```

The lines in this file indicate that the `toggle` argument `crmod`, whose default value is "off" (or FALSE), should be enabled when connecting to the system `weirdhost`. In addition, the value of the environment variable `PRINTER` should be exported to all systems. In this case, the `DEFAULT` keyword is used in place of the host name.

### FILES

- `$HOME/.telnetrc`

### SEE ALSO
telnet(1), `in.telnetd(1M)`, `environ(5)`
The term file is compiled from terminfo(4) source files using tic(1M). Compiled files are organized in a directory hierarchy under the first letter of each terminal name. For example, the vt100 file would have the pathname 
/usr/lib/terminfo/v/vt100. The default directory is 
/usr/share/lib/terminfo. Synonyms for the same terminal are implemented by multiple links to the same compiled file.

The format has been chosen so that it is the same on all hardware. An 8-bit byte is assumed, but no assumptions about byte ordering or sign extension are made. Thus, these binary terminfo files can be transported to other hardware with 8-bit bytes.

Short integers are stored in two 8-bit bytes. The first byte contains the least significant 8 bits of the value, and the second byte contains the most significant 8 bits. (Thus, the value represented is 256*second+first.) The value −1 is represented by 0377, 0377, and the value −2 is represented by 0376, 0377; other negative values are illegal. The −1 generally means that a capability is missing from this terminal. The −2 means that the capability has been cancelled in the terminfo source and also is to be considered missing.

The compiled file is created from the source file descriptions of the terminals (see the -I option of infocmp) by using the terminfo compiler, tic, and read by the routine setupterm (see curses(3CURSES)). The file is divided into six parts in the following order: the header, terminal names, boolean flags, numbers, strings, and string table.

The header section begins the file six short integers in the format described below. These integers are:

1. the magic number (octal 0432);
2. the size, in bytes, of the names section;
3. the number of bytes in the boolean section
4. the number of short integers in the numbers section;
5. the number of offsets (short integers) in the strings section;
6. the size, in bytes, of the string table.

The terminal name section comes next. It contains the first line of the terminfo description, listing the various names for the terminal, separated by the bar (|) character (see term(5)). The section is terminated with an ASCII NUL character.

The terminal name section is followed by the Boolean section, number section, string section, and string table.

The boolean flags section consists of one byte for each flag. This byte is either 0 or 1 as the flag is present or absent. The value of 2 means that the flag has been cancelled. The capabilities are in the same order as the file <term.h>.
term(4)
Between the boolean flags section and the number section, a null byte is inserted, if
necessary, to ensure that the number section begins on an even byte offset. All short
integers are aligned on a short word boundary.
The numbers section is similar to the boolean flags section. Each capability takes up
two bytes, and is stored as a short integer. If the value represented is −1 or −2, the
capability is taken to be missing.
The strings section is also similar. Each capability is stored as a short integer, in the
format above. A value of −1 or −2 means the capability is missing. Otherwise, the
value is taken as an offset from the beginning of the string table. Special characters in
^X or \c notation are stored in their interpreted form, not the printing representation.
Padding information ($<nn>) and parameter information (%x) are stored intact in
uninterpreted form.
The final section is the string table. It contains all the values of string capabilities
referenced in the string section. Each string is null terminated.
Note that it is possible for setupterm to expect a different set of capabilities than are
actually present in the file. Either the database may have been updated since
setupterm has been recompiled (resulting in extra unrecognized entries in the file) or
the program may have been recompiled more recently than the database was updated
(resulting in missing entries). The routine setupterm must be prepared for both
possibilities—this is why the numbers and sizes are included. Also, new capabilities
must always be added at the end of the lists of boolean, number, and string
capabilities.
As an example, here is terminal information on the AT&T Model 37 KSR terminal as
output by the infocmp -I tty37 command:
37|tty37|AT&T model 37 teletype,
hc, os, xon,
bel=^G, cr=\r, cub1=\b, cud1=\n, cuu1=\E7, hd=\E9,
hu=\E8, ind=\n,

The following is an octal dump of the corresponding term file, produced by the od
-c /usr/share/lib/terminfo/t/tty37 command:
0000000
0000020
0000040
0000060
0000100
0000120
0000140
0000160
0000200
0000220
*
0000520
0000540
0000560

470

032 001
t
y
3
3
7
\0 \0 \0
001 \0 \0
377 377 377
\0 377
377 377
"
377 377
0
377 377 377

\0
7
t
001
\0
377
377
\0
\0
377

032
|
e
\0
\0
377
377
377
377
377

\0
A
l
\0
\0
377
377
377
377
377

013
T
e
\0
377
377
377
377
377
377

\0
&
t
\0
377
377
377
377
377
377

021
T
y
\0
377
377
377
(
377
377

001
p
\0
377
377
377
\0
377
377

3
m
e
\0
377
377
377
377
377
377

\0
o
\0
001
377
377
377
377
377
377

3
d
\0
\0
377
377
377
377
377

7
e
\0
\0
377
377
377
377
\0
377

|
l
\0
\0
377
&
377
377
377
377

t
\0
\0
377
\0
377
377
377
377

377 377 377 377 377 377 377 377 377 377 377 377 377 377
$ \0
377 377 377 377 377 377 377 377 377 377 377 377 377 377
* \0

man pages section 4: File Formats • Last Revised 3 Jul 1996


Some limitations: total compiled entries cannot exceed 4096 bytes; all entries in the name field cannot exceed 128 bytes.

FILES
/usr/share/lib/terminfo/?/* compiled terminal description database
/usr/include/term.h  terminfo header
/usr/xpg4/include/term.h  X/Open Curses terminfo header

SEE ALSO  infocmp(1M), curses(3CURSES), curses(3XCURSES), terminfo(4), term(5)
terminfo – terminal and printer capability database

/usr/share/lib/terminfo/?/*

terminfo is a database that describes the capabilities of devices such as terminals and printers. Devices are described in terminfo source files by specifying a set of capabilities, by quantifying certain aspects of the device, and by specifying character sequences that affect particular results. This database is often used by screen oriented applications such as vi and curses-based programs, as well as by some system commands such as ls and more. This usage allows them to work with a variety of devices without changes to the programs.

termino descriptions are located in the directory pointed to by the environment variable TERMINFO or in /usr/share/lib/terminfo. terminfo descriptions are generated by tic(1M).

terminfo source files consist of one or more device descriptions. Each description consists of a header (beginning in column 1) and one or more lines that list the features for that particular device. Every line in a terminfo source file must end in a comma (,). Every line in a terminfo source file except the header must be indented with one or more white spaces (either spaces or tabs).

Entries in terminfo source files consist of a number of comma-separated fields. White space after each comma is ignored. Embedded commas must be escaped by using a backslash. Each device entry has the following format:

alias₁ | alias₂ | . . . | aliasₙ | fullname,
        | capability₁, | capability₂, |
        | .            |
        | .            |
        | capabilityₙ, |

The first line, commonly referred to as the header line, must begin in column one and must contain at least two aliases separated by vertical bars. The last field in the header line must be the long name of the device and it may contain any string. Alias names must be unique in the terminfo database and they must conform to system file naming conventions (see tic(1M)); they cannot, for example, contain white space or slashes.

Every device must be assigned a name, such as "vt100". Device names (except the long name) should be chosen using the following conventions. The name should not contain hyphens because hyphens are reserved for use when adding suffixes that indicate special modes.

These special modes may be modes that the hardware can be in, or user preferences. To assign a special mode to a particular device, append a suffix consisting of a hyphen and an indicator of the mode to the device name. For example, the -w suffix means
"wide mode"; when specified, it allows for a width of 132 columns instead of the standard 80 columns. Therefore, if you want to use a "vt100" device set to wide mode, name the device "vt100-w." Use the following suffixes where possible.

<table>
<thead>
<tr>
<th>Suffix</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>-w</td>
<td>Wide mode (more than 80 columns)</td>
<td>5410-w</td>
</tr>
<tr>
<td>-am</td>
<td>With auto. margins (usually default)</td>
<td>vt100-am</td>
</tr>
<tr>
<td>-nam</td>
<td>Without automatic margins</td>
<td>vt100-nam</td>
</tr>
<tr>
<td>-n</td>
<td>Number of lines on the screen</td>
<td>2300-40</td>
</tr>
<tr>
<td>-na</td>
<td>No arrow keys (leave them in local)</td>
<td>c100-na</td>
</tr>
<tr>
<td>-np</td>
<td>Number of pages of memory</td>
<td>c100-4p</td>
</tr>
<tr>
<td>-rv</td>
<td>Reverse video</td>
<td>4415-rv</td>
</tr>
</tbody>
</table>

The terminfo reference manual page is organized in two sections:

- PART 1: DEVICE CAPABILITIES
- PART 2: PRINTER CAPABILITIES

Capabilities in terminfo are of three types: Boolean capabilities (which show that a device has or does not have a particular feature), numeric capabilities (which quantify particular features of a device), and string capabilities (which provide sequences that can be used to perform particular operations on devices).

In the following table, a Variable is the name by which a C programmer accesses a capability (at the terminfo level). A Capname is the short name for a capability specified in the terminfo source file. It is used by a person updating the source file and by the tput command. A Termcap Code is a two-letter sequence that corresponds to the termcap capability name. (Note that termcap is no longer supported.)

Capability names have no real length limit, but an informal limit of five characters has been adopted to keep them short. Whenever possible, capability names are chosen to be the same as or similar to those specified by the ANSI X3.64-1979 standard. Semantics are also intended to match those of the ANSI standard.

All string capabilities listed below may have padding specified, with the exception of those used for input. Input capabilities, listed under the Strings section in the following tables, have names beginning with key_. The #i symbol in the description field of the following tables refers to the ith parameter.
### TABLE 1 Booleans

<table>
<thead>
<tr>
<th>Variable</th>
<th>Name</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto_left_margin</td>
<td>bw</td>
<td>bw</td>
<td><code>cub1</code> wraps from column 0 to last column</td>
</tr>
<tr>
<td>auto_right_margin</td>
<td>am</td>
<td>am</td>
<td>Terminal has automatic margins</td>
</tr>
<tr>
<td>back_color_erase</td>
<td>bce</td>
<td>be</td>
<td>Screen erased with background color</td>
</tr>
<tr>
<td>can_change</td>
<td>ccc</td>
<td>cc</td>
<td>Terminal can re-define existing color</td>
</tr>
<tr>
<td>ceol_standout_glitch</td>
<td>xhp</td>
<td>xs</td>
<td>Standout not erased by overwriting (<code>hp</code>)</td>
</tr>
<tr>
<td>col_addr_glitch</td>
<td>xhpa</td>
<td>YA</td>
<td>Only positive motion for <code>hpa/\hpa</code> caps</td>
</tr>
<tr>
<td>cpi_changes_res</td>
<td>cpix</td>
<td>YF</td>
<td>Changing character pitch changes resolution</td>
</tr>
<tr>
<td>cr_cancels_micro_mode</td>
<td>crxm</td>
<td>YB</td>
<td>Using <code>cr</code> turns off micro mode</td>
</tr>
<tr>
<td>dest_tabs_magic_smso</td>
<td>xt</td>
<td>xt</td>
<td>Destructive tabs, magic <code>smso</code> char (<code>1061</code>)</td>
</tr>
<tr>
<td>eat_newline_glitch</td>
<td>xenl</td>
<td>xn</td>
<td>Newline ignored after 80 columns (Concept)</td>
</tr>
<tr>
<td>erase_overstrike</td>
<td>eo</td>
<td>eo</td>
<td>Can erase overstrikes with a blank</td>
</tr>
<tr>
<td>generic_type</td>
<td>gn</td>
<td>gn</td>
<td>Generic line type (for example, <code>dialup</code>, <code>switch</code>)</td>
</tr>
<tr>
<td>hard_copy</td>
<td>hc</td>
<td>hc</td>
<td>Hardcopy terminal</td>
</tr>
<tr>
<td>hard_cursor</td>
<td>chts</td>
<td>HC</td>
<td>Cursor is hard to see</td>
</tr>
<tr>
<td>has_meta_key</td>
<td>km</td>
<td>km</td>
<td>Has a meta key (shift, sets parity bit)</td>
</tr>
<tr>
<td>has_print_wheel</td>
<td>daisy</td>
<td>YC</td>
<td>Printer needs operator to change character set</td>
</tr>
<tr>
<td>has_status_line</td>
<td>hs</td>
<td>hs</td>
<td>Has extra &quot;status line&quot;</td>
</tr>
<tr>
<td>hue_lightness_saturation</td>
<td>hls</td>
<td>hl</td>
<td>Terminal uses only HLS color notation (Tektronix)</td>
</tr>
<tr>
<td>insert_null_glitch</td>
<td>in</td>
<td>in</td>
<td>Insert mode distinguishes nulls</td>
</tr>
<tr>
<td>lpi_changes_res</td>
<td>lpix</td>
<td>YG</td>
<td>Changing line pitch changes resolution</td>
</tr>
<tr>
<td>memory_above</td>
<td>da</td>
<td>da</td>
<td>Display may be retained above the screen</td>
</tr>
<tr>
<td>memory_below</td>
<td>db</td>
<td>db</td>
<td>Display may be retained below the screen</td>
</tr>
</tbody>
</table>
### TABLE 1 Booleans (Continued)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Name</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>move_insert_mode</td>
<td>mir</td>
<td>mi</td>
<td>Safe to move while in insert mode</td>
</tr>
<tr>
<td>move_standout_mode</td>
<td>msgr</td>
<td>ms</td>
<td>Safe to move in standout modes</td>
</tr>
<tr>
<td>needs_xon_xoff</td>
<td>nxon</td>
<td>nx</td>
<td>Padding won’t work, xon/xoff required</td>
</tr>
<tr>
<td>no_esc_ctlc</td>
<td>xsb</td>
<td>xb</td>
<td>Beehive (f1=escape, f2=ctrl C)</td>
</tr>
<tr>
<td>no_pad_char</td>
<td>npc</td>
<td>NP</td>
<td>Pad character doesn’t exist</td>
</tr>
<tr>
<td>non_dest_scroll_region</td>
<td>ndscr</td>
<td>ND</td>
<td>Scrolling region is nondestructive</td>
</tr>
<tr>
<td>non_rev_rmcup</td>
<td>nrrmc</td>
<td>NR</td>
<td>smcup does not reverse smcup</td>
</tr>
<tr>
<td>over_strike</td>
<td>os</td>
<td>os</td>
<td>Terminal overstrikes on hard-copy</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>terminal</td>
</tr>
<tr>
<td>prtr_silent</td>
<td>mc5i</td>
<td>5i</td>
<td>Printer won’t echo on screen</td>
</tr>
<tr>
<td>row_addr_glitch</td>
<td>xvpa</td>
<td>YD</td>
<td>Only positive motion for vpa/mvpa caps</td>
</tr>
<tr>
<td>semi_auto_right_margin</td>
<td>sam</td>
<td>YE</td>
<td>Printing in last column causes cr</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>status_line_esc_ok</td>
<td>eslok</td>
<td>es</td>
<td>Escape can be used on the status line</td>
</tr>
<tr>
<td>tilde_glitch</td>
<td>hz</td>
<td>hz</td>
<td>Hazeltine; can’t print tilde (~)</td>
</tr>
<tr>
<td>transparent_underline</td>
<td>ul</td>
<td>ul</td>
<td>Underline character overstrikes</td>
</tr>
<tr>
<td>xon_xoff</td>
<td>xon</td>
<td>xo</td>
<td>Terminal uses xon/xoff handshaking</td>
</tr>
</tbody>
</table>

### TABLE 2 Numbers

<table>
<thead>
<tr>
<th>Variable</th>
<th>Name</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit_image_entwining</td>
<td>bitwin</td>
<td>Yo</td>
<td>Number of passes for each bit-map row</td>
</tr>
<tr>
<td>bit_image_type</td>
<td>bitype</td>
<td>Yp</td>
<td>Type of bit image device</td>
</tr>
<tr>
<td>buffer_capacity</td>
<td>bufsz</td>
<td>Ya</td>
<td>Number of bytes buffered before printing</td>
</tr>
<tr>
<td>buttons</td>
<td>btns</td>
<td>BT</td>
<td>Number of buttons on the mouse</td>
</tr>
<tr>
<td>columns</td>
<td>cols</td>
<td>co</td>
<td>Number of columns in a line</td>
</tr>
<tr>
<td>dot_horz_spacing</td>
<td>spinh</td>
<td>Yc</td>
<td>Spacing of dots horizontally in dots per inch</td>
</tr>
<tr>
<td>dot_vert_spacing</td>
<td>spinv</td>
<td>Yb</td>
<td>Spacing of pins vertically in pins per inch</td>
</tr>
<tr>
<td>Variable</td>
<td>Name</td>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>------</td>
<td>------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>init_tabs</td>
<td>it</td>
<td>it</td>
<td>Tabs initially every # spaces</td>
</tr>
<tr>
<td>label_height</td>
<td>lh</td>
<td>lh</td>
<td>Number of rows in each label</td>
</tr>
<tr>
<td>label_width</td>
<td>lw</td>
<td>lw</td>
<td>Number of columns in each label</td>
</tr>
<tr>
<td>lines</td>
<td>lines</td>
<td>li</td>
<td>Number of lines on a screen or a page</td>
</tr>
<tr>
<td>lines_of_memory</td>
<td>lm</td>
<td>lm</td>
<td>Lines of memory if &gt; lines; 0 means varies</td>
</tr>
<tr>
<td>max_attributes</td>
<td>ma</td>
<td>ma</td>
<td>Maximum combined video attributes</td>
</tr>
<tr>
<td>magic_cookie_glitch</td>
<td>xmc</td>
<td>sg</td>
<td>Number of blank characters left by $\texttt{smso}$ or $\texttt{rmso}$</td>
</tr>
<tr>
<td>max_colors</td>
<td>colors</td>
<td>Co</td>
<td>Maximum number of colors on the screen</td>
</tr>
<tr>
<td>max_micro_address</td>
<td>maddr</td>
<td>Yd</td>
<td>Maximum value in $\texttt{micro...address}$</td>
</tr>
<tr>
<td>max_micro_jump</td>
<td>mjump</td>
<td>Ye</td>
<td>Maximum value in $\texttt{parm...micro}$</td>
</tr>
<tr>
<td>max_pairs</td>
<td>pairs</td>
<td>pa</td>
<td>Maximum number of color-pairs on the screen</td>
</tr>
<tr>
<td>maximum_windows</td>
<td>wnum</td>
<td>MW</td>
<td>Maximum number of definable windows</td>
</tr>
<tr>
<td>micro_char_size</td>
<td>mcs</td>
<td>Yf</td>
<td>Character step size when in micro mode</td>
</tr>
<tr>
<td>micro_line_size</td>
<td>mls</td>
<td>Yg</td>
<td>Line step size when in micro mode</td>
</tr>
<tr>
<td>no_color_video</td>
<td>ncv</td>
<td>NC</td>
<td>Video attributes that can’t be used with colors</td>
</tr>
<tr>
<td>num_labels</td>
<td>nlab</td>
<td>NI</td>
<td>Number of labels on screen (start at 1)</td>
</tr>
<tr>
<td>number_of_pins</td>
<td>npins</td>
<td>Yh</td>
<td>Number of pins in print-head</td>
</tr>
<tr>
<td>output_res_char</td>
<td>orc</td>
<td>Yi</td>
<td>Horizontal resolution in units per character</td>
</tr>
</tbody>
</table>
### TABLE 2 Numbers (Continued)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Name</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>output_res_line</td>
<td>orl</td>
<td>Yj</td>
<td>Vertical resolution in units per line</td>
</tr>
<tr>
<td>output_res_horz_inch</td>
<td>orhi</td>
<td>Yk</td>
<td>Horizontal resolution in units per inch</td>
</tr>
<tr>
<td>output_res_vert_inch</td>
<td>orvi</td>
<td>Yl</td>
<td>Vertical resolution in units per inch</td>
</tr>
<tr>
<td>padding_baud_rate</td>
<td>pb</td>
<td>pb</td>
<td>Lowest baud rate where padding needed</td>
</tr>
<tr>
<td>print_rate</td>
<td>cps</td>
<td>Ym</td>
<td>Print rate in characters per second</td>
</tr>
<tr>
<td>virtual_terminal</td>
<td>vt</td>
<td>vt</td>
<td>Virtual terminal number (system)</td>
</tr>
<tr>
<td>wide_char_size</td>
<td>widcs</td>
<td>Yn</td>
<td>Character step size when in double wide mode</td>
</tr>
<tr>
<td>width_status_line</td>
<td>wsl</td>
<td>ws</td>
<td>Number of columns in status line</td>
</tr>
</tbody>
</table>

### TABLE 3 Strings

<table>
<thead>
<tr>
<th>Variable</th>
<th>Name</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>acs_chars</td>
<td>acsc</td>
<td>ac</td>
<td>Graphic charset pairs aAbBcC</td>
</tr>
<tr>
<td>alt_scancode_esc</td>
<td>scesa</td>
<td>S8</td>
<td>Alternate escape for scancode emulation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(default is for vt100)</td>
</tr>
<tr>
<td>back_tab</td>
<td>cbt</td>
<td>bt</td>
<td>Back tab</td>
</tr>
<tr>
<td>bell</td>
<td>bel</td>
<td>bl</td>
<td>Audible signal (bell)</td>
</tr>
<tr>
<td>bit_image_carriage_return</td>
<td>bicr</td>
<td>Yv</td>
<td>Move to beginning of same row (use tparm)</td>
</tr>
<tr>
<td>bit_image_newline</td>
<td>binel</td>
<td>Zz</td>
<td>Move to next row of the bit image (use tparm)</td>
</tr>
<tr>
<td>bit_image_repeat</td>
<td>birep</td>
<td>Zy</td>
<td>Repeat bit-image cell #1 #2 times (use tparm)</td>
</tr>
<tr>
<td>carriage_return</td>
<td>cr</td>
<td>cr</td>
<td>Carriage return</td>
</tr>
<tr>
<td>Variable</td>
<td>Name</td>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------</td>
<td>------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>change_char_pitch</td>
<td>cpi</td>
<td>ZA</td>
<td>Change number of characters per inch</td>
</tr>
<tr>
<td>change_line_pitch</td>
<td>lpi</td>
<td>ZB</td>
<td>Change number of lines per inch</td>
</tr>
<tr>
<td>change_res_horz</td>
<td>chr</td>
<td>ZC</td>
<td>Change horizontal resolution</td>
</tr>
<tr>
<td>change_res_vert</td>
<td>cvr</td>
<td>ZD</td>
<td>Change vertical resolution</td>
</tr>
<tr>
<td>change_scroll_region</td>
<td>csr</td>
<td>cs</td>
<td>Change to lines #1 through #2 (vt100)</td>
</tr>
<tr>
<td>char_padding</td>
<td>rmp</td>
<td>rP</td>
<td>Like <code>ip</code> but when in replace mode</td>
</tr>
<tr>
<td>char_set_names</td>
<td>csnm</td>
<td>Zy</td>
<td>List of character set names</td>
</tr>
<tr>
<td>clear_all_tabs</td>
<td>tbc</td>
<td>ct</td>
<td>Clear all tab stops</td>
</tr>
<tr>
<td>clear_margins</td>
<td>mgc</td>
<td>MC</td>
<td>Clear all margins (top, bottom, and sides)</td>
</tr>
<tr>
<td>clear_screen</td>
<td>clear</td>
<td>cl</td>
<td>Clear screen and home cursor</td>
</tr>
<tr>
<td>clr_bol</td>
<td>el1</td>
<td>cb</td>
<td>Clear to beginning of line, inclusive</td>
</tr>
<tr>
<td>clr_eol</td>
<td>el</td>
<td>ce</td>
<td>Clear to end of line</td>
</tr>
<tr>
<td>clr_eos</td>
<td>ed</td>
<td>cd</td>
<td>Clear to end of display</td>
</tr>
<tr>
<td>code_set_init</td>
<td>csin</td>
<td>ci</td>
<td>Init sequence for multiple codesets</td>
</tr>
<tr>
<td>color_names</td>
<td>colrm</td>
<td>Yw</td>
<td>Give name for color #1</td>
</tr>
<tr>
<td>column_address</td>
<td>bpa</td>
<td>ch</td>
<td>Horizontal position absolute</td>
</tr>
<tr>
<td>command_character</td>
<td>cmdch</td>
<td>CC</td>
<td>Terminal settable cmd character in prototype</td>
</tr>
<tr>
<td>create_window</td>
<td>cwin</td>
<td>CW</td>
<td>Define win #1 to go from #2,#3 to #4,#5</td>
</tr>
<tr>
<td>cursor_address</td>
<td>cup</td>
<td>cm</td>
<td>Move to row #1 col #2</td>
</tr>
<tr>
<td>cursor_down</td>
<td>cud1</td>
<td>do</td>
<td>Down one line</td>
</tr>
<tr>
<td>cursor_home</td>
<td>home</td>
<td>ho</td>
<td>Home cursor (if no <code>cup</code>)</td>
</tr>
<tr>
<td>cursor_invisible</td>
<td>civis</td>
<td>vi</td>
<td>Make cursor invisible</td>
</tr>
<tr>
<td>cursor_left</td>
<td>cub1</td>
<td>le</td>
<td>Move left one space</td>
</tr>
<tr>
<td>cursor_mem_address</td>
<td>mrcup</td>
<td>CM</td>
<td>Memory relative cursor addressing</td>
</tr>
<tr>
<td>Variable</td>
<td>Name</td>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------</td>
<td>------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>cursor_normal</td>
<td>cnorm</td>
<td>ve</td>
<td>Make cursor appear normal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(undo vs/vi)</td>
</tr>
<tr>
<td>cursor_right</td>
<td>cufl</td>
<td>nd</td>
<td>Non-destructive space (cursor or carriage right)</td>
</tr>
<tr>
<td>cursor_to_ll</td>
<td>ll</td>
<td>ll</td>
<td>Last line, first column (if no cup)</td>
</tr>
<tr>
<td>cursor_up</td>
<td>cuu1</td>
<td>up</td>
<td>Upline (cursor up)</td>
</tr>
<tr>
<td>cursor_visible</td>
<td>cvvis</td>
<td>vs</td>
<td>Make cursor very visible</td>
</tr>
<tr>
<td>define_bit_image_region</td>
<td>defbi</td>
<td>Yx</td>
<td>Define rectangular bit-image region</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(use tparm)</td>
</tr>
<tr>
<td>define_char</td>
<td>defc</td>
<td>ZE</td>
<td>Define a character in a character set*</td>
</tr>
<tr>
<td>delete_character</td>
<td>dch1</td>
<td>dc</td>
<td>Delete character</td>
</tr>
<tr>
<td>delete_line</td>
<td>dl1</td>
<td>dl</td>
<td>Delete line</td>
</tr>
<tr>
<td>device_type</td>
<td>devt</td>
<td>dv</td>
<td>Indicate language/codeset support</td>
</tr>
<tr>
<td>dial_phone</td>
<td>dial</td>
<td>DI</td>
<td>Dial phone number #1</td>
</tr>
<tr>
<td>dis_status_line</td>
<td>dsl</td>
<td>ds</td>
<td>Disable status line</td>
</tr>
<tr>
<td>display_clock</td>
<td>dclk</td>
<td>DK</td>
<td>Display time-of-day clock</td>
</tr>
<tr>
<td>display_pc_char</td>
<td>dispc</td>
<td>SI</td>
<td>Display PC character</td>
</tr>
<tr>
<td>down_half_line</td>
<td>hd</td>
<td>hd</td>
<td>Half-line down (forward 1/2 linefeed)</td>
</tr>
<tr>
<td>ena_acs</td>
<td>enacs</td>
<td>eA</td>
<td>Enable alternate character set</td>
</tr>
<tr>
<td>end_bit_image_region</td>
<td>endbi</td>
<td>Yy</td>
<td>End a bit-image region (use tparm)</td>
</tr>
<tr>
<td>enter_alt_charset_mode</td>
<td>smacs</td>
<td>as</td>
<td>Start alternate character set</td>
</tr>
<tr>
<td>enter_am_mode</td>
<td>smam</td>
<td>SA</td>
<td>Turn on automatic margins</td>
</tr>
<tr>
<td>enter_blink_mode</td>
<td>blink</td>
<td>mb</td>
<td>Turn on blinking</td>
</tr>
<tr>
<td>enter_bold_mode</td>
<td>bold</td>
<td>md</td>
<td>Turn on bold (extra bright) mode</td>
</tr>
<tr>
<td>enter_ca_mode</td>
<td>smcup</td>
<td>ti</td>
<td>String to begin programs that use cup</td>
</tr>
<tr>
<td>enter_delete_mode</td>
<td>smdc</td>
<td>dm</td>
<td>Delete mode (enter)</td>
</tr>
<tr>
<td>Variable</td>
<td>Name</td>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------</td>
<td>------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>enter_dim_mode</td>
<td>dim</td>
<td>mh</td>
<td>Turn on half-bright mode</td>
</tr>
<tr>
<td>enter_doublewide_mode</td>
<td>swidm</td>
<td>ZF</td>
<td>Enable double wide printing</td>
</tr>
<tr>
<td>enter_draft_quality</td>
<td>sdrfq</td>
<td>ZG</td>
<td>Set draft quality print mode</td>
</tr>
<tr>
<td>enter_insert_mode</td>
<td>smir</td>
<td>im</td>
<td>Insert mode (enter)</td>
</tr>
<tr>
<td>enter_italics_mode</td>
<td>sitm</td>
<td>ZH</td>
<td>Enable italics</td>
</tr>
<tr>
<td>enter_leftward_mode</td>
<td>slim</td>
<td>ZI</td>
<td>Enable leftward carriage motion</td>
</tr>
<tr>
<td>enter_micro_mode</td>
<td>smicm</td>
<td>ZJ</td>
<td>Enable micro motion capabilities</td>
</tr>
<tr>
<td>enter_near_letter_quality</td>
<td>snlq</td>
<td>ZK</td>
<td>Set near-letter quality print</td>
</tr>
<tr>
<td>enter_normal_quality</td>
<td>snrmq</td>
<td>ZL</td>
<td>Set normal quality print</td>
</tr>
<tr>
<td>enter_pc_charset_mode</td>
<td>smpch</td>
<td>S2</td>
<td>Enter PC character display mode</td>
</tr>
<tr>
<td>enter_protected_mode</td>
<td>prot</td>
<td>mp</td>
<td>Turn on protected mode</td>
</tr>
<tr>
<td>enter_reverse_mode</td>
<td>rev</td>
<td>mr</td>
<td>Turn on reverse video mode</td>
</tr>
<tr>
<td>enter_scancode_mode</td>
<td>smsc</td>
<td>S4</td>
<td>Enter PC scancode mode</td>
</tr>
<tr>
<td>enter_secure_mode</td>
<td>invis</td>
<td>mk</td>
<td>Turn on blank mode (characters invisible)</td>
</tr>
<tr>
<td>enter_shadow_mode</td>
<td>sshm</td>
<td>ZM</td>
<td>Enable shadow printing</td>
</tr>
<tr>
<td>enter_standout_mode</td>
<td>smso</td>
<td>so</td>
<td>Begin standout mode</td>
</tr>
<tr>
<td>enter_subscript_mode</td>
<td>ssusbm</td>
<td>ZN</td>
<td>Enable subscript printing</td>
</tr>
<tr>
<td>enter_superscript_mode</td>
<td>ssupm</td>
<td>ZO</td>
<td>Enable superscript printing</td>
</tr>
<tr>
<td>enter_underline_mode</td>
<td>smul</td>
<td>us</td>
<td>Start underscore mode</td>
</tr>
<tr>
<td>enter_upward_mode</td>
<td>sum</td>
<td>ZP</td>
<td>Enable upward carriage motion</td>
</tr>
<tr>
<td>enter_xon_mode</td>
<td>smxton</td>
<td>SX</td>
<td>Turn on xon/xoff handshaking</td>
</tr>
<tr>
<td>erase_chars</td>
<td>ech</td>
<td>ec</td>
<td>Erase #1 characters</td>
</tr>
<tr>
<td>exit_alt_charset_mode</td>
<td>rmacs</td>
<td>ae</td>
<td>End alternate character set</td>
</tr>
<tr>
<td>exit_am_mode</td>
<td>rmam</td>
<td>RA</td>
<td>Turn off automatic margins</td>
</tr>
<tr>
<td>exit_attribute_mode</td>
<td>sgr0</td>
<td>me</td>
<td>Turn off all attributes</td>
</tr>
<tr>
<td>Variable</td>
<td>Name</td>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------</td>
<td>------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>exit_ca_mode</td>
<td>rmcup</td>
<td>te</td>
<td>String to end programs that use cup</td>
</tr>
<tr>
<td>exit_delete_mode</td>
<td>rmdec</td>
<td>ed</td>
<td>End delete mode</td>
</tr>
<tr>
<td>exit_doublewide_mode</td>
<td>rwdm</td>
<td>ZQ</td>
<td>Disable double wide printing</td>
</tr>
<tr>
<td>exit_insert_mode</td>
<td>rmir</td>
<td>ei</td>
<td>End insert mode</td>
</tr>
<tr>
<td>exit_italics_mode</td>
<td>ritm</td>
<td>ZR</td>
<td>Disable italics</td>
</tr>
<tr>
<td>exit_leftward_mode</td>
<td>rlm</td>
<td>ZS</td>
<td>Enable rightward (normal)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>carriage motion</td>
</tr>
<tr>
<td>exit_micro_mode</td>
<td>rmicm</td>
<td>ZT</td>
<td>Disable micro motion capabilities</td>
</tr>
<tr>
<td>exit_pc_charset_mode</td>
<td>rmpch</td>
<td>S3</td>
<td>Disable PC character display mode</td>
</tr>
<tr>
<td>exit_scancode_mode</td>
<td>rmsc</td>
<td>S5</td>
<td>Disable PC scancode mode</td>
</tr>
<tr>
<td>exit_shadow_mode</td>
<td>rshm</td>
<td>ZU</td>
<td>Disable shadow printing</td>
</tr>
<tr>
<td>exit_standout_mode</td>
<td>rmso</td>
<td>se</td>
<td>End standout mode</td>
</tr>
<tr>
<td>exit_subscript_mode</td>
<td>rsubm</td>
<td>ZV</td>
<td>Disable subscript printing</td>
</tr>
<tr>
<td>exit_superscript_mode</td>
<td>rsupm</td>
<td>ZW</td>
<td>Disable superscript printing</td>
</tr>
<tr>
<td>exit_underline_mode</td>
<td>rmul</td>
<td>ue</td>
<td>End underscore mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>carriage motion</td>
</tr>
<tr>
<td>exit_upward_mode</td>
<td>rum</td>
<td>ZX</td>
<td>Enable downward (normal)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>carriage motion</td>
</tr>
<tr>
<td>exit_xon_mode</td>
<td>rmxon</td>
<td>RX</td>
<td>Turn off xon/xoff handshaking</td>
</tr>
<tr>
<td>fixed_pause</td>
<td>pause</td>
<td>PA</td>
<td>Pause for 2-3 seconds</td>
</tr>
<tr>
<td>flash_hook</td>
<td>hook</td>
<td>fh</td>
<td>Flash the switch hook</td>
</tr>
<tr>
<td>flash_screen</td>
<td>flash</td>
<td>vb</td>
<td>Visible bell (may not move cursor)</td>
</tr>
<tr>
<td>form_feed</td>
<td>ff</td>
<td>ff</td>
<td>Hardcopy terminal page eject</td>
</tr>
<tr>
<td>from_status_line</td>
<td>fsl</td>
<td>fs</td>
<td>Return from status line</td>
</tr>
<tr>
<td>get_mouse</td>
<td>getm</td>
<td>Gm</td>
<td>Curses should get button events</td>
</tr>
<tr>
<td>goto_window</td>
<td>wingo</td>
<td>WG</td>
<td>Go to window #1</td>
</tr>
<tr>
<td>hangup</td>
<td>hup</td>
<td>HU</td>
<td>Hang-up phone</td>
</tr>
<tr>
<td>init_1string</td>
<td>is1</td>
<td>i1</td>
<td>Terminal or printer initialization string</td>
</tr>
</tbody>
</table>
The “key_” strings are sent by specific keys. The “key_” descriptions include the macro, defined in `<curses.h>`, for the code returned by the `curses` routine `getch` when the key is pressed (see `curs_getch(3CURSES)`).

### TABLE 3 Strings (Continued)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Name</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>init_2string</td>
<td>is2</td>
<td>is</td>
<td>Terminal or printer initialization string</td>
</tr>
<tr>
<td>init_3string</td>
<td>is3</td>
<td>i3</td>
<td>Terminal or printer initialization string</td>
</tr>
<tr>
<td>init_file</td>
<td>if</td>
<td>if</td>
<td>Name of initialization file</td>
</tr>
<tr>
<td>init_prog</td>
<td>iprog</td>
<td>iP</td>
<td>Path name of program for initialization</td>
</tr>
<tr>
<td>initialize_color</td>
<td>initc</td>
<td>lc</td>
<td>Initialize the definition of color</td>
</tr>
<tr>
<td>initialize_pair</td>
<td>initp</td>
<td>lp</td>
<td>Initialize color-pair</td>
</tr>
<tr>
<td>insert_character</td>
<td>ich1</td>
<td>ic</td>
<td>Insert character</td>
</tr>
<tr>
<td>insert_line</td>
<td>il1</td>
<td>al</td>
<td>Add new blank line</td>
</tr>
<tr>
<td>insert_padding</td>
<td>ip</td>
<td>ip</td>
<td>Insert pad after character inserted</td>
</tr>
</tbody>
</table>

### TABLE 4 key_ Strings

<table>
<thead>
<tr>
<th>Variable</th>
<th>Name</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key_a1</td>
<td>ka1</td>
<td>K1</td>
<td>KEY_A1, upper left of keypad</td>
</tr>
<tr>
<td>key_a3</td>
<td>ka3</td>
<td>K3</td>
<td>KEY_A3, upper right of keypad</td>
</tr>
<tr>
<td>key_b2</td>
<td>kb2</td>
<td>K2</td>
<td>KEY_B2, center of keypad</td>
</tr>
<tr>
<td>key_backspace</td>
<td>kbs</td>
<td>kb</td>
<td>KEY_BACKSPACE, sent by backspace</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>key</td>
</tr>
<tr>
<td>key_beg</td>
<td>kbeg</td>
<td>@1</td>
<td>KEY_BEG, sent by beg(inning) key</td>
</tr>
<tr>
<td>key_btab</td>
<td>kbt</td>
<td>kB</td>
<td>KEY_BTAB, sent by back-tab key</td>
</tr>
<tr>
<td>key_c1</td>
<td>kc1</td>
<td>K4</td>
<td>KEY_C1, lower left of keypad</td>
</tr>
<tr>
<td>key_c3</td>
<td>kc3</td>
<td>K5</td>
<td>KEY_C3, lower right of keypad</td>
</tr>
<tr>
<td>key_cancel</td>
<td>kcn</td>
<td>@2</td>
<td>KEYCANCEL, sent by cancel key</td>
</tr>
<tr>
<td>key_catab</td>
<td>ktbc</td>
<td>ka</td>
<td>KEY_CATAB, sent by clear-all-tabs key</td>
</tr>
<tr>
<td>Variable</td>
<td>Name</td>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>------</td>
<td>------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>key_clear</td>
<td>kclr</td>
<td>kC</td>
<td>KEY_CLEAR, sent by clear-screen or erase key</td>
</tr>
<tr>
<td>key_close</td>
<td>kclo</td>
<td>@3</td>
<td>KEY_CLOSE, sent by close key</td>
</tr>
<tr>
<td>key_command</td>
<td>kcmd</td>
<td>@4</td>
<td>KEY_COMMAND, sent by cmd (command) key</td>
</tr>
<tr>
<td>key_copy</td>
<td>kcpy</td>
<td>@5</td>
<td>KEY_COPY, sent by copy key</td>
</tr>
<tr>
<td>key_create</td>
<td>kcrt</td>
<td>@6</td>
<td>KEY_CREATE, sent by create key</td>
</tr>
<tr>
<td>key_ctab</td>
<td>kctab</td>
<td>kt</td>
<td>KEY_CTAB, sent by clear-tab key</td>
</tr>
<tr>
<td>key_dc</td>
<td>kdch1</td>
<td>kD</td>
<td>KEY_DC, sent by delete-character key</td>
</tr>
<tr>
<td>key_dl</td>
<td>kdl1</td>
<td>kL</td>
<td>KEY_DL, sent by delete-line key</td>
</tr>
<tr>
<td>key_down</td>
<td>kcud1</td>
<td>kd</td>
<td>KEY_DOWN, sent by terminal down-arrow key</td>
</tr>
<tr>
<td>key_eic</td>
<td>krmir</td>
<td>kM</td>
<td>KEY_EIC, sent by rmir or smir in insert mode</td>
</tr>
<tr>
<td>key_end</td>
<td>kend</td>
<td>@7</td>
<td>KEY_END, sent by end key</td>
</tr>
<tr>
<td>key_enter</td>
<td>kent</td>
<td>@8</td>
<td>KEY_ENTER, sent by enter/send key</td>
</tr>
<tr>
<td>key_eol</td>
<td>kel</td>
<td>kE</td>
<td>KEY_EOL, sent by clear-to-end-of-line key</td>
</tr>
<tr>
<td>key_eos</td>
<td>ked</td>
<td>kS</td>
<td>KEY_EOS, sent by clear-to-end-of-screen key</td>
</tr>
<tr>
<td>key_exit</td>
<td>kext</td>
<td>@9</td>
<td>KEY_EXIT, sent by exit key</td>
</tr>
<tr>
<td>key_f0</td>
<td>kf0</td>
<td>k0</td>
<td>KEY_F(0), sent by function key f0</td>
</tr>
<tr>
<td>key_f1</td>
<td>kf1</td>
<td>k1</td>
<td>KEY_F(1), sent by function key f1</td>
</tr>
<tr>
<td>key_f2</td>
<td>kf2</td>
<td>k2</td>
<td>KEY_F(2), sent by function key f2</td>
</tr>
<tr>
<td>key_f3</td>
<td>kf3</td>
<td>k3</td>
<td>KEY_F(3), sent by function key f3</td>
</tr>
<tr>
<td>key_f4</td>
<td>kf4</td>
<td>k4</td>
<td>KEY_F(4), sent by function key f4</td>
</tr>
<tr>
<td>key_f5</td>
<td>kf5</td>
<td>k5</td>
<td>KEY_F(5), sent by function key f5</td>
</tr>
<tr>
<td>Variable</td>
<td>Name</td>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>key_f6</td>
<td>kf6</td>
<td>k6</td>
<td>\texttt{KEY_F(6)}, sent by function key f6</td>
</tr>
<tr>
<td>key_f7</td>
<td>kf7</td>
<td>k7</td>
<td>\texttt{KEY_F(7)}, sent by function key f7</td>
</tr>
<tr>
<td>key_f8</td>
<td>kf8</td>
<td>k8</td>
<td>\texttt{KEY_F(8)}, sent by function key f8</td>
</tr>
<tr>
<td>key_f9</td>
<td>kf9</td>
<td>k9</td>
<td>\texttt{KEY_F(9)}, sent by function key f9</td>
</tr>
<tr>
<td>key_f10</td>
<td>kf10</td>
<td>k;</td>
<td>\texttt{KEY_F(10)}, sent by function key f10</td>
</tr>
<tr>
<td>key_f11</td>
<td>kf11</td>
<td>F1</td>
<td>\texttt{KEY_F(11)}, sent by function key f11</td>
</tr>
<tr>
<td>key_f12</td>
<td>kf12</td>
<td>F2</td>
<td>\texttt{KEY_F(12)}, sent by function key f12</td>
</tr>
<tr>
<td>key_f13</td>
<td>kf13</td>
<td>F3</td>
<td>\texttt{KEY_F(13)}, sent by function key f13</td>
</tr>
<tr>
<td>key_f14</td>
<td>kf14</td>
<td>F4</td>
<td>\texttt{KEY_F(14)}, sent by function key f14</td>
</tr>
<tr>
<td>key_f15</td>
<td>kf15</td>
<td>F5</td>
<td>\texttt{KEY_F(15)}, sent by function key f15</td>
</tr>
<tr>
<td>key_f16</td>
<td>kf16</td>
<td>F6</td>
<td>\texttt{KEY_F(16)}, sent by function key f16</td>
</tr>
<tr>
<td>key_f17</td>
<td>kf17</td>
<td>F7</td>
<td>\texttt{KEY_F(17)}, sent by function key f17</td>
</tr>
<tr>
<td>key_f18</td>
<td>kf18</td>
<td>F8</td>
<td>\texttt{KEY_F(18)}, sent by function key f18</td>
</tr>
<tr>
<td>key_f19</td>
<td>kf19</td>
<td>F9</td>
<td>\texttt{KEY_F(19)}, sent by function key f19</td>
</tr>
<tr>
<td>key_f20</td>
<td>kf20</td>
<td>FA</td>
<td>\texttt{KEY_F(20)}, sent by function key f20</td>
</tr>
<tr>
<td>key_f21</td>
<td>kf21</td>
<td>FB</td>
<td>\texttt{KEY_F(21)}, sent by function key f21</td>
</tr>
<tr>
<td>key_f22</td>
<td>kf22</td>
<td>FC</td>
<td>\texttt{KEY_F(22)}, sent by function key f22</td>
</tr>
<tr>
<td>key_f23</td>
<td>kf23</td>
<td>FD</td>
<td>\texttt{KEY_F(23)}, sent by function key f23</td>
</tr>
<tr>
<td>key_f24</td>
<td>kf24</td>
<td>FE</td>
<td>\texttt{KEY_F(24)}, sent by function key f24</td>
</tr>
<tr>
<td>Variable</td>
<td>Name</td>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
<td>------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>key_f25</td>
<td>kf25</td>
<td>FF</td>
<td>KEY_F(25), sent by function key f25</td>
</tr>
<tr>
<td>key_f26</td>
<td>kf26</td>
<td>FG</td>
<td>KEY_F(26), sent by function key f26</td>
</tr>
<tr>
<td>key_f27</td>
<td>kf27</td>
<td>FH</td>
<td>KEY_F(27), sent by function key f27</td>
</tr>
<tr>
<td>key_f28</td>
<td>kf28</td>
<td>FI</td>
<td>KEY_F(28), sent by function key f28</td>
</tr>
<tr>
<td>key_f29</td>
<td>kf29</td>
<td>FJ</td>
<td>KEY_F(29), sent by function key f29</td>
</tr>
<tr>
<td>key_f30</td>
<td>kf30</td>
<td>FK</td>
<td>KEY_F(30), sent by function key f30</td>
</tr>
<tr>
<td>key_f31</td>
<td>kf31</td>
<td>FL</td>
<td>KEY_F(31), sent by function key f31</td>
</tr>
<tr>
<td>key_f32</td>
<td>kf32</td>
<td>FM</td>
<td>KEY_F(32), sent by function key f32</td>
</tr>
<tr>
<td>key_f33</td>
<td>kf33</td>
<td>FN</td>
<td>KEY_F(34), sent by function key f34</td>
</tr>
<tr>
<td>key_f34</td>
<td>kf34</td>
<td>FO</td>
<td>KEY_F(35), sent by function key f35</td>
</tr>
<tr>
<td>key_f35</td>
<td>kf35</td>
<td>FP</td>
<td>KEY_F(36), sent by function key f36</td>
</tr>
<tr>
<td>key_f36</td>
<td>kf36</td>
<td>FQ</td>
<td>KEY_F(37), sent by function key f37</td>
</tr>
<tr>
<td>key_f37</td>
<td>kf37</td>
<td>FR</td>
<td>KEY_F(38), sent by function key f38</td>
</tr>
<tr>
<td>key_f38</td>
<td>kf38</td>
<td>FS</td>
<td>KEY_F(39), sent by function key f39</td>
</tr>
<tr>
<td>key_f39</td>
<td>kf39</td>
<td>FT</td>
<td>KEY_F(40), sent by function key fB0</td>
</tr>
<tr>
<td>key_fB0</td>
<td>kf40</td>
<td>FU</td>
<td>KEY_F(41), sent by function key fB1</td>
</tr>
<tr>
<td>key_fB1</td>
<td>kf41</td>
<td>FW</td>
<td>KEY_F(42), sent by function key fB2</td>
</tr>
</tbody>
</table>
### TABLE 4 keyStrings (Continued)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Name</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key_fB3</td>
<td>kf43</td>
<td>FX</td>
<td>KEY_P(43), sent by function key fB3</td>
</tr>
<tr>
<td>key_fB4</td>
<td>kf44</td>
<td>FY</td>
<td>KEY_P(44), sent by function key fB4</td>
</tr>
<tr>
<td>key_fB5</td>
<td>kf45</td>
<td>FZ</td>
<td>KEY_P(45), sent by function key fB5</td>
</tr>
<tr>
<td>key_fB6</td>
<td>kf46</td>
<td>Fa</td>
<td>KEY_P(46), sent by function key fB6</td>
</tr>
<tr>
<td>key_fB7</td>
<td>kf47</td>
<td>Fb</td>
<td>KEY_P(47), sent by function key fB7</td>
</tr>
<tr>
<td>key_fB8</td>
<td>kf48</td>
<td>Fc</td>
<td>KEY_P(48), sent by function key fB8</td>
</tr>
<tr>
<td>key_fB9</td>
<td>kf49</td>
<td>Fd</td>
<td>KEY_P(49), sent by function key fB9</td>
</tr>
<tr>
<td>key_f50</td>
<td>kf50</td>
<td>Fe</td>
<td>KEY_P(50), sent by function key f50</td>
</tr>
<tr>
<td>key_f51</td>
<td>kf51</td>
<td>Ff</td>
<td>KEY_P(51), sent by function key f51</td>
</tr>
<tr>
<td>key_f52</td>
<td>kf52</td>
<td>Fg</td>
<td>KEY_P(52), sent by function key f52</td>
</tr>
<tr>
<td>key_f53</td>
<td>kf53</td>
<td>Fh</td>
<td>KEY_P(53), sent by function key f53</td>
</tr>
<tr>
<td>key_f54</td>
<td>kf54</td>
<td>Fi</td>
<td>KEY_P(54), sent by function key f54</td>
</tr>
<tr>
<td>key_f55</td>
<td>kf55</td>
<td>Fj</td>
<td>KEY_P(55), sent by function key f55</td>
</tr>
<tr>
<td>key_f56</td>
<td>kf56</td>
<td>Fk</td>
<td>KEY_P(56), sent by function key f56</td>
</tr>
<tr>
<td>key_f57</td>
<td>kf57</td>
<td>Fl</td>
<td>KEY_P(57), sent by function key f57</td>
</tr>
<tr>
<td>key_f58</td>
<td>kf58</td>
<td>Fm</td>
<td>KEY_P(58), sent by function key f58</td>
</tr>
<tr>
<td>key_f59</td>
<td>kf59</td>
<td>Fn</td>
<td>KEY_P(59), sent by function key f59</td>
</tr>
<tr>
<td>key_f60</td>
<td>kf60</td>
<td>Fo</td>
<td>KEY_P(60), sent by function key f60</td>
</tr>
<tr>
<td>Variable</td>
<td>Name</td>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------</td>
<td>------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>key_f61</td>
<td>kf61</td>
<td>Fp</td>
<td>KEY_F(61), sent by function key f61</td>
</tr>
<tr>
<td>key_f62</td>
<td>kf62</td>
<td>Fq</td>
<td>KEY_F(62), sent by function key f62</td>
</tr>
<tr>
<td>key_f63</td>
<td>kf63</td>
<td>Fr</td>
<td>KEY_F(63), sent by function key f63</td>
</tr>
<tr>
<td>key_find</td>
<td>kfind</td>
<td>@0</td>
<td>KEY_FIND, sent by find key</td>
</tr>
<tr>
<td>key_help</td>
<td>khlp</td>
<td>%1</td>
<td>KEY_HELP, sent by help key</td>
</tr>
<tr>
<td>key_home</td>
<td>khome</td>
<td>kh</td>
<td>KEY_HOME, sent by home key</td>
</tr>
<tr>
<td>key_ic</td>
<td>kich1</td>
<td>kI</td>
<td>KEY_IC, sent by ins-char/enter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ins-mode key</td>
</tr>
<tr>
<td>key_il</td>
<td>kil1</td>
<td>kA</td>
<td>KEY_IL, sent by insert-line key</td>
</tr>
<tr>
<td>key_left</td>
<td>kcub1</td>
<td>kl</td>
<td>KEY_LEFT, sent by terminal left-arrow</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>key</td>
</tr>
<tr>
<td>key_ll</td>
<td>kll</td>
<td>kH</td>
<td>KEY_LL, sent by home-down key</td>
</tr>
<tr>
<td>key_mark</td>
<td>kmrk</td>
<td>%2</td>
<td>KEY_MARK, sent by mark key</td>
</tr>
<tr>
<td>key_message</td>
<td>kmsg</td>
<td>%3</td>
<td>KEY_MESSAGE, sent by message key</td>
</tr>
<tr>
<td>key_mouse</td>
<td>kmous</td>
<td>Km</td>
<td>0631, Mouse event has occurred</td>
</tr>
<tr>
<td>key_move</td>
<td>kmov</td>
<td>%4</td>
<td>KEY_MOVE, sent by move key</td>
</tr>
<tr>
<td>key_next</td>
<td>knxt</td>
<td>%5</td>
<td>KEY_NEXT, sent by next-object key</td>
</tr>
<tr>
<td>key_npage</td>
<td>knp</td>
<td>kN</td>
<td>KEY_NPAGE, sent by next-page key</td>
</tr>
<tr>
<td>key_open</td>
<td>kopn</td>
<td>%6</td>
<td>KEY_OPEN, sent by open key</td>
</tr>
<tr>
<td>key_options</td>
<td>kopt</td>
<td>%7</td>
<td>KEY_OPTIONS, sent by options key</td>
</tr>
<tr>
<td>key_ppage</td>
<td>kpp</td>
<td>kP</td>
<td>KEY_PPAGE, sent by previous-page key</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>key</td>
</tr>
<tr>
<td>key_previous</td>
<td>kprv</td>
<td>%8</td>
<td>KEY_PREVIOUS, sent by previous-object key</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>key</td>
</tr>
<tr>
<td>key_print</td>
<td>kprt</td>
<td>%9</td>
<td>KEY_PRINT, sent by print or copy key</td>
</tr>
<tr>
<td>Variable</td>
<td>Name</td>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------</td>
<td>------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>key_redo</td>
<td>krdo</td>
<td>%0</td>
<td>KEY_REDO, sent by redo key</td>
</tr>
<tr>
<td>key_reference</td>
<td>kref</td>
<td>&amp;1</td>
<td>KEY_REFERENCE, sent by reference key</td>
</tr>
<tr>
<td>key_refresh</td>
<td>krfr</td>
<td>&amp;2</td>
<td>KEY_REFRESH, sent by refresh key</td>
</tr>
<tr>
<td>key_replace</td>
<td>krpl</td>
<td>&amp;3</td>
<td>KEY_REPLACE, sent by replace key</td>
</tr>
<tr>
<td>key_restart</td>
<td>krst</td>
<td>&amp;4</td>
<td>KEY_RESTART, sent by restart key</td>
</tr>
<tr>
<td>key_resume</td>
<td>kres</td>
<td>&amp;5</td>
<td>KEY_RESUME, sent by resume key</td>
</tr>
<tr>
<td>key_right</td>
<td>kcuf1</td>
<td>kr</td>
<td>KEY_RIGHT, sent by terminal right-arrow key</td>
</tr>
<tr>
<td>key_save</td>
<td>ksav</td>
<td>&amp;6</td>
<td>KEY_SAVE, sent by save key</td>
</tr>
<tr>
<td>key_sbeg</td>
<td>kBEG</td>
<td>&amp;9</td>
<td>KEY_SBEG, sent by shifted beginning key</td>
</tr>
<tr>
<td>key_scancel</td>
<td>kCAN</td>
<td>&amp;0</td>
<td>KEY_SCANCEL, sent by shifted cancel key</td>
</tr>
<tr>
<td>key_scommand</td>
<td>kCMD</td>
<td>*1</td>
<td>KEY_SCOMMAND, sent by shifted command key</td>
</tr>
<tr>
<td>key_scopy</td>
<td>kCPY</td>
<td>*2</td>
<td>KEY_SCOPY, sent by shifted copy key</td>
</tr>
<tr>
<td>key_screate</td>
<td>kCRT</td>
<td>*3</td>
<td>KEY_SCREATE, sent by shifted create key</td>
</tr>
<tr>
<td>key_sdc</td>
<td>kDC</td>
<td>*4</td>
<td>KEY_SDC, sent by shifted delete-char key</td>
</tr>
<tr>
<td>key_sdl</td>
<td>kDL</td>
<td>*5</td>
<td>KEY_SDL, sent by shifted delete-line key</td>
</tr>
<tr>
<td>key_select</td>
<td>kslt</td>
<td>*6</td>
<td>KEY_SELECT, sent by select key</td>
</tr>
<tr>
<td>key_send</td>
<td>kEND</td>
<td>*7</td>
<td>KEY_SEND, sent by shifted end key</td>
</tr>
<tr>
<td>key_seol</td>
<td>kEOL</td>
<td>*8</td>
<td>KEY_SEOL, sent by shifted clear-line key</td>
</tr>
<tr>
<td>key_seexit</td>
<td>kEXT</td>
<td>*9</td>
<td>KEY_SEXIT, sent by shifted exit key</td>
</tr>
</tbody>
</table>
### Table 4: key_strings (Continued)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Name</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key_sf</td>
<td>kind</td>
<td>kF</td>
<td>KEY_SF, sent by scroll-forward/down key</td>
</tr>
<tr>
<td>key_sfind</td>
<td>kFND</td>
<td>*0</td>
<td>KEY_SFIN, sent by shifted find key</td>
</tr>
<tr>
<td>key_shelp</td>
<td>kHLP</td>
<td>#1</td>
<td>KEY_SHELP, sent by shifted help key</td>
</tr>
<tr>
<td>key_shome</td>
<td>kHOM</td>
<td>#2</td>
<td>KEY_SHOME, sent by shifted home key</td>
</tr>
<tr>
<td>key_sic</td>
<td>kIC</td>
<td>#3</td>
<td>KEY_SIC, sent by shifted input key</td>
</tr>
<tr>
<td>key_sleft</td>
<td>kLFT</td>
<td>#4</td>
<td>KEY_SLEFT, sent by shifted left-arrow key</td>
</tr>
<tr>
<td>key_smessage</td>
<td>kMSG</td>
<td>%a</td>
<td>KEY_SMESSAGE, sent by shifted message key</td>
</tr>
<tr>
<td>key_smov</td>
<td>kMOV</td>
<td>%b</td>
<td>KEY_SMOVE, sent by shifted move key</td>
</tr>
<tr>
<td>key_snex</td>
<td>kNXT</td>
<td>%c</td>
<td>KEY_SNEXT, sent by shifted next key</td>
</tr>
<tr>
<td>key_sop</td>
<td>kOPT</td>
<td>%d</td>
<td>KEY_SOPTIONS, sent by shifted options key</td>
</tr>
<tr>
<td>key_spre</td>
<td>kPRV</td>
<td>%e</td>
<td>KEY_SPREVIOUS, sent by shifted prev key</td>
</tr>
<tr>
<td>key_spri</td>
<td>kPRT</td>
<td>%f</td>
<td>KEY_SPRINT, sent by shifted print key</td>
</tr>
<tr>
<td>key_sr</td>
<td>kri</td>
<td>kR</td>
<td>KEY_SR, sent by scroll-backward/up key</td>
</tr>
<tr>
<td>key_sredo</td>
<td>kRDO</td>
<td>%g</td>
<td>KEY_SREDO, sent by shifted redo key</td>
</tr>
<tr>
<td>key_srep</td>
<td>kRPL</td>
<td>%h</td>
<td>KEY_SREPLACE, sent by shifted replace</td>
</tr>
</tbody>
</table>

File Formats 489
<table>
<thead>
<tr>
<th>Variable</th>
<th>Name</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key_sright</td>
<td>kRIT</td>
<td>%i</td>
<td>KEY_SRIGHT, sent by shifted right-arrow key</td>
</tr>
<tr>
<td>key_srsxme</td>
<td>kRES</td>
<td>%j</td>
<td>KEY_SRSUME, sent by shifted resume</td>
</tr>
<tr>
<td>key_ssave</td>
<td>kSAV</td>
<td>!1</td>
<td>KEY_SSAVE, sent by shifted save key</td>
</tr>
<tr>
<td>key_ssuspend</td>
<td>kSPD</td>
<td>!2</td>
<td>KEY_SSUSPEND, sent by shifted suspend key</td>
</tr>
<tr>
<td>key_stab</td>
<td>khts</td>
<td>kT</td>
<td>KEY_STAB, sent by set-tab key</td>
</tr>
<tr>
<td>key_sundo</td>
<td>kUND</td>
<td>!3</td>
<td>KEY_SUNDO, sent by shifted undo key</td>
</tr>
<tr>
<td>key_suspend</td>
<td>kspd</td>
<td>&amp;7</td>
<td>KEY_SUSPEND, sent by suspend key</td>
</tr>
<tr>
<td>key_undo</td>
<td>kund</td>
<td>&amp;8</td>
<td>KEY_UNDO, sent by undo key</td>
</tr>
<tr>
<td>key_up</td>
<td>kcui</td>
<td>ku</td>
<td>KEY_UP, sent by terminal up-arrow key</td>
</tr>
<tr>
<td>keypad_local</td>
<td>rmkx</td>
<td>ke</td>
<td>Out of “keypad-transmit” mode</td>
</tr>
<tr>
<td>keypad_xmit</td>
<td>smkx</td>
<td>ks</td>
<td>Put terminal in “keypad-transmit” mode</td>
</tr>
<tr>
<td>lab_f0</td>
<td>lf0</td>
<td>l0</td>
<td>Labels on function key f0 if not f0</td>
</tr>
<tr>
<td>lab_f1</td>
<td>lf1</td>
<td>l1</td>
<td>Labels on function key f1 if not f1</td>
</tr>
<tr>
<td>lab_f2</td>
<td>lf2</td>
<td>l2</td>
<td>Labels on function key f2 if not f2</td>
</tr>
<tr>
<td>lab_f3</td>
<td>lf3</td>
<td>l3</td>
<td>Labels on function key f3 if not f3</td>
</tr>
<tr>
<td>lab_f8</td>
<td>lfd</td>
<td>l4</td>
<td>Labels on function key f8 if not f8</td>
</tr>
<tr>
<td>lab_f5</td>
<td>lf5</td>
<td>l5</td>
<td>Labels on function key f5 if not f5</td>
</tr>
<tr>
<td>lab_f6</td>
<td>lf6</td>
<td>l6</td>
<td>Labels on function key f6 if not f6</td>
</tr>
<tr>
<td>lab_f7</td>
<td>lf7</td>
<td>l7</td>
<td>Labels on function key f7 if not f7</td>
</tr>
<tr>
<td>lab_f8</td>
<td>lf8</td>
<td>l8</td>
<td>Labels on function key f8 if not f8</td>
</tr>
<tr>
<td>lab_f9</td>
<td>lf9</td>
<td>l9</td>
<td>Labels on function key f9 if not f9</td>
</tr>
</tbody>
</table>
### TABLE 4 key_Strings (Continued)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Name</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lab_f10</td>
<td>lf10</td>
<td>la</td>
<td>Labels on function key f10 if not f10</td>
</tr>
<tr>
<td>label_format</td>
<td>fl</td>
<td>Lf</td>
<td>Label format</td>
</tr>
<tr>
<td>label_off</td>
<td>rmln</td>
<td>LF</td>
<td>Turn off soft labels</td>
</tr>
<tr>
<td>label_on</td>
<td>smln</td>
<td>LO</td>
<td>Turn on soft labels</td>
</tr>
<tr>
<td>meta_off</td>
<td>rmm</td>
<td>mo</td>
<td>Turn off &quot;meta mode&quot;</td>
</tr>
<tr>
<td>meta_on</td>
<td>smm</td>
<td>mm</td>
<td>Turn on &quot;meta mode&quot; (8th bit)</td>
</tr>
<tr>
<td>micro_column_address</td>
<td>mhpa</td>
<td>ZY</td>
<td>Like column_address for micro adjustment</td>
</tr>
<tr>
<td>micro_down</td>
<td>mcud1</td>
<td>ZZ</td>
<td>Like cursor_down for micro adjustment</td>
</tr>
<tr>
<td>micro_left</td>
<td>mcub1</td>
<td>Za</td>
<td>Like cursor_left for micro adjustment</td>
</tr>
<tr>
<td>micro_right</td>
<td>mcuf1</td>
<td>Zb</td>
<td>Like cursor_right for micro adjustment</td>
</tr>
<tr>
<td>micro_row_address</td>
<td>mvpa</td>
<td>Zc</td>
<td>Like row_address for micro adjustment</td>
</tr>
<tr>
<td>micro_up</td>
<td>mcuu1</td>
<td>Zd</td>
<td>Like cursor_up for micro adjustment</td>
</tr>
<tr>
<td>mouse_info</td>
<td>minfo</td>
<td>Mi</td>
<td>Mouse status information</td>
</tr>
<tr>
<td>newline</td>
<td>nel</td>
<td>nw</td>
<td>Newline (behaves like cr followed by lf)</td>
</tr>
<tr>
<td>order_of_pins</td>
<td>porder</td>
<td>Ze</td>
<td>Matches software bits to print-head pins</td>
</tr>
<tr>
<td>orig_colors</td>
<td>oc</td>
<td>oc</td>
<td>Set all color(-pair)s to the original ones</td>
</tr>
<tr>
<td>orig_pair</td>
<td>op</td>
<td>op</td>
<td>Set default color-pair to the original one</td>
</tr>
<tr>
<td>pad_char</td>
<td>pad</td>
<td>pc</td>
<td>Pad character (rather than null)</td>
</tr>
<tr>
<td>parm_dch</td>
<td>dch</td>
<td>DC</td>
<td>Delete #1 chars</td>
</tr>
<tr>
<td>parm_delete_line</td>
<td>dl</td>
<td>DL</td>
<td>Delete #1 lines</td>
</tr>
<tr>
<td>parm_down_cursor</td>
<td>cud</td>
<td>DO</td>
<td>Move down #1 lines.</td>
</tr>
<tr>
<td>Variable</td>
<td>Name</td>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------</td>
<td>------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>parm_down_micro</td>
<td>mcud</td>
<td>Zf</td>
<td>Like parm_down_cursor for micro adjust.</td>
</tr>
<tr>
<td>parm_ich</td>
<td>ich</td>
<td>IC</td>
<td>Insert #1 blank chars</td>
</tr>
<tr>
<td>parm_index</td>
<td>indn</td>
<td>SF</td>
<td>Scroll forward #1 lines</td>
</tr>
<tr>
<td>parm_insert_line</td>
<td>il</td>
<td>AL</td>
<td>Add #1 new blank lines</td>
</tr>
<tr>
<td>parm_left_cursor</td>
<td>cub</td>
<td>LE</td>
<td>Move cursor left #1 spaces</td>
</tr>
<tr>
<td>parm_left_micro</td>
<td>mcub</td>
<td>Zg</td>
<td>Like parm_left_cursor for micro adjust.</td>
</tr>
<tr>
<td>parm_right_cursor</td>
<td>cuf</td>
<td>RI</td>
<td>Move right #1 spaces</td>
</tr>
<tr>
<td>parm_right_micro</td>
<td>mcuf</td>
<td>Zh</td>
<td>Like parm_right_cursor for micro adjust.</td>
</tr>
<tr>
<td>parm_rindex</td>
<td>rin</td>
<td>SR</td>
<td>Scroll backward #1 lines</td>
</tr>
<tr>
<td>parm_up_cursor</td>
<td>cuu</td>
<td>UP</td>
<td>Move cursor up #1 lines</td>
</tr>
<tr>
<td>parm_up_micro</td>
<td>mcuu</td>
<td>Zi</td>
<td>Like parm_up_cursor for micro adjust.</td>
</tr>
<tr>
<td>pc_term_options</td>
<td>pctrm</td>
<td>S6</td>
<td>PC terminal options</td>
</tr>
<tr>
<td>pkey_key</td>
<td>pfkey</td>
<td>pk</td>
<td>Prog funct key #1 to type string #2</td>
</tr>
<tr>
<td>pkey_local</td>
<td>pfloc</td>
<td>pl</td>
<td>Prog funct key #1 to execute string #2</td>
</tr>
<tr>
<td>pkey_plab</td>
<td>pfxl</td>
<td>xl</td>
<td>Prog key #1 to xmit string #2 and show string #3</td>
</tr>
<tr>
<td>pkey_xmit</td>
<td>pfx</td>
<td>px</td>
<td>Prog funct key #1 to xmit string #2</td>
</tr>
<tr>
<td>plab_norm</td>
<td>pln</td>
<td>pn</td>
<td>Prog label #1 to show string #2</td>
</tr>
<tr>
<td>print_screen</td>
<td>mc0</td>
<td>ps</td>
<td>Print contents of the screen</td>
</tr>
<tr>
<td>prtr_non</td>
<td>mc5p</td>
<td>pO</td>
<td>Turn on the printer for #1 bytes</td>
</tr>
<tr>
<td>prtr_off</td>
<td>mc4</td>
<td>pf</td>
<td>Turn off the printer</td>
</tr>
<tr>
<td>prtr_on</td>
<td>mc5</td>
<td>po</td>
<td>Turn on the printer</td>
</tr>
<tr>
<td>pulse</td>
<td>pulse</td>
<td>PU</td>
<td>Select pulse dialing</td>
</tr>
<tr>
<td>Variable</td>
<td>Name</td>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------</td>
<td>------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>quick_dial</td>
<td>qdial</td>
<td>QD</td>
<td>Dial phone number #1, without progress detection</td>
</tr>
<tr>
<td>remove_clock</td>
<td>rmclk</td>
<td>RC</td>
<td>Remove time-of-day clock</td>
</tr>
<tr>
<td>repeat_char</td>
<td>rep</td>
<td>rp</td>
<td>Repeat char #1 #2 times</td>
</tr>
<tr>
<td>req_for_input</td>
<td>rfi</td>
<td>RF</td>
<td>Send next input char (for ptys)</td>
</tr>
<tr>
<td>req_mouse_pos</td>
<td>reqmp</td>
<td>RQ</td>
<td>Request mouse position report</td>
</tr>
<tr>
<td>reset_1string</td>
<td>rs1</td>
<td>r1</td>
<td>Reset terminal completely to sane modes</td>
</tr>
<tr>
<td>reset_2string</td>
<td>rs2</td>
<td>r2</td>
<td>Reset terminal completely to sane modes</td>
</tr>
<tr>
<td>reset_3string</td>
<td>rs3</td>
<td>r3</td>
<td>Reset terminal completely to sane modes</td>
</tr>
<tr>
<td>reset_file</td>
<td>rf</td>
<td>rf</td>
<td>Name of file containing reset string</td>
</tr>
<tr>
<td>restore_cursor</td>
<td>rc</td>
<td>rc</td>
<td>Restore cursor to position of last sc</td>
</tr>
<tr>
<td>row_address</td>
<td>vpa</td>
<td>cv</td>
<td>Vertical position absolute</td>
</tr>
<tr>
<td>save_cursor</td>
<td>sc</td>
<td>sc</td>
<td>Save cursor position</td>
</tr>
<tr>
<td>scancode_escape</td>
<td>scesc</td>
<td>S7</td>
<td>Escape for scancode emulation</td>
</tr>
<tr>
<td>scroll_forward</td>
<td>ind</td>
<td>sf</td>
<td>Scroll text up</td>
</tr>
<tr>
<td>scroll_reverse</td>
<td>ri</td>
<td>sr</td>
<td>Scroll text down</td>
</tr>
<tr>
<td>select_char_set</td>
<td>scs</td>
<td>Zj</td>
<td>Select character set</td>
</tr>
<tr>
<td>set0_des_seq</td>
<td>s0ds</td>
<td>s0</td>
<td>Shift into codeset 0 (EUC set 0, ASCII)</td>
</tr>
<tr>
<td>set1_des_seq</td>
<td>s1ds</td>
<td>s1</td>
<td>Shift into codeset 1</td>
</tr>
<tr>
<td>set2_des_seq</td>
<td>s2ds</td>
<td>s2</td>
<td>Shift into codeset 2</td>
</tr>
<tr>
<td>set3_des_seq</td>
<td>s3ds</td>
<td>s3</td>
<td>Shift into codeset 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Attributes #1-#6</td>
</tr>
<tr>
<td>set_a_background</td>
<td>setab</td>
<td>AB</td>
<td>Set background color using ANSI escape</td>
</tr>
<tr>
<td>set_a_foreground</td>
<td>setaf</td>
<td>AF</td>
<td>Set foreground color using ANSI escape</td>
</tr>
<tr>
<td>set_attributes</td>
<td>sgr</td>
<td>sa</td>
<td>Define the video attributes #1-#9</td>
</tr>
<tr>
<td>Variable</td>
<td>Name</td>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------</td>
<td>------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>set_background</td>
<td>setb</td>
<td>Sb</td>
<td>Set current background color</td>
</tr>
<tr>
<td>set_bottom_margin</td>
<td>smgb</td>
<td>Zk</td>
<td>Set bottom margin at current line</td>
</tr>
<tr>
<td>set_bottom_margin_parm</td>
<td>smgbp</td>
<td>Zl</td>
<td>Set bottom margin at line #1 or #2 lines from bottom</td>
</tr>
<tr>
<td>set_clock</td>
<td>sclk</td>
<td>SC</td>
<td>Set time-of-day clock</td>
</tr>
<tr>
<td>set_color_band</td>
<td>setcolor</td>
<td>Yz</td>
<td>Change to ribbon color #1</td>
</tr>
<tr>
<td>set_color_pair</td>
<td>scp</td>
<td>sp</td>
<td>Set current color-pair</td>
</tr>
<tr>
<td>set_foreground</td>
<td>setf</td>
<td>Sf</td>
<td>Set current foreground color1</td>
</tr>
<tr>
<td>set_left_margin</td>
<td>smgl</td>
<td>ML</td>
<td>Set left margin at current line</td>
</tr>
<tr>
<td>set_left_margin_parm</td>
<td>smglp</td>
<td>Zm</td>
<td>Set left (right) margin at column #1 (#2)</td>
</tr>
<tr>
<td>set_lr_margin</td>
<td>smgr</td>
<td>MR</td>
<td>Sets both left and right margins</td>
</tr>
<tr>
<td>set_page_length</td>
<td>slines</td>
<td>YZ</td>
<td>Set page length to #1 lines (use tparm)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>of an inch</td>
</tr>
<tr>
<td>set_right_margin</td>
<td>smgr</td>
<td>MR</td>
<td>Set right margin at current column</td>
</tr>
<tr>
<td>set_right_margin_parm</td>
<td>smgrp</td>
<td>Zn</td>
<td>Set right margin at column #1</td>
</tr>
<tr>
<td>set_tab</td>
<td>hts</td>
<td>st</td>
<td>Set a tab in all rows, current column</td>
</tr>
<tr>
<td>set_tb_margin</td>
<td>smgtb</td>
<td>MT</td>
<td>Sets both top and bottom margins</td>
</tr>
<tr>
<td>set_top_margin</td>
<td>smgt</td>
<td>Zo</td>
<td>Set top margin at current line</td>
</tr>
<tr>
<td>set_top_margin_parm</td>
<td>smgtp</td>
<td>Zp</td>
<td>Set top (bottom) margin at line #1 (#2)</td>
</tr>
<tr>
<td>set_window</td>
<td>wind</td>
<td>wi</td>
<td>Current window is lines #1-#2 cols #3-#4</td>
</tr>
<tr>
<td>start_bit_image</td>
<td>sbim</td>
<td>Zq</td>
<td>Start printing bit image graphics</td>
</tr>
<tr>
<td>start_char_set_def</td>
<td>scsd</td>
<td>Zr</td>
<td>Start definition of a character set</td>
</tr>
<tr>
<td>stop_bit_image</td>
<td>rbim</td>
<td>Zs</td>
<td>End printing bit image graphics</td>
</tr>
<tr>
<td>stop_char_set_def</td>
<td>rcsd</td>
<td>Zt</td>
<td>End definition of a character set</td>
</tr>
<tr>
<td>subscript_characters</td>
<td>subcs</td>
<td>Zu</td>
<td>List of “subscript-able” characters</td>
</tr>
<tr>
<td>superscript_characters</td>
<td>supcs</td>
<td>Zv</td>
<td>List of “superscript-able” characters</td>
</tr>
</tbody>
</table>
### TABLE 4 key_Strings (Continued)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Name</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tab</td>
<td>ht</td>
<td>ta</td>
<td>Tab to next 8-space hardware tab stop</td>
</tr>
<tr>
<td>these_cause_cr</td>
<td>docr</td>
<td>Zw</td>
<td>Printing any of these chars causes cr</td>
</tr>
<tr>
<td>to_status_line</td>
<td>tsl</td>
<td>ts</td>
<td>Go to status line, col #1</td>
</tr>
<tr>
<td>tone</td>
<td>tone</td>
<td>TO</td>
<td>Select touch tone dialing</td>
</tr>
<tr>
<td>user0</td>
<td>u0</td>
<td>u0</td>
<td>User string 0</td>
</tr>
<tr>
<td>user1</td>
<td>u1</td>
<td>u1</td>
<td>User string 1</td>
</tr>
<tr>
<td>user2</td>
<td>u2</td>
<td>u2</td>
<td>User string 2</td>
</tr>
<tr>
<td>user3</td>
<td>u3</td>
<td>u3</td>
<td>User string 3</td>
</tr>
<tr>
<td>user4</td>
<td>u4</td>
<td>u4</td>
<td>User string 4</td>
</tr>
<tr>
<td>user5</td>
<td>u5</td>
<td>u5</td>
<td>User string 5</td>
</tr>
<tr>
<td>user6</td>
<td>u6</td>
<td>u6</td>
<td>User string 6</td>
</tr>
<tr>
<td>user7</td>
<td>u7</td>
<td>u7</td>
<td>User string 7</td>
</tr>
<tr>
<td>user8</td>
<td>u8</td>
<td>u8</td>
<td>User string 8</td>
</tr>
<tr>
<td>user9</td>
<td>u9</td>
<td>u9</td>
<td>User string 9</td>
</tr>
<tr>
<td>underline_char</td>
<td>uc</td>
<td>uc</td>
<td>Underscore one char and move past it</td>
</tr>
<tr>
<td>up_half_line</td>
<td>hu</td>
<td>hu</td>
<td>Half-line up (reverse 1/2 linefeed)</td>
</tr>
<tr>
<td>wait_tone</td>
<td>wait</td>
<td>WA</td>
<td>Wait for dial tone</td>
</tr>
<tr>
<td>xoff_character</td>
<td>xoffc</td>
<td>XF</td>
<td>X-off character</td>
</tr>
<tr>
<td>xon_character</td>
<td>xonc</td>
<td>XN</td>
<td>X-on character</td>
</tr>
<tr>
<td>zero_motion</td>
<td>zerom</td>
<td>Zx</td>
<td>No motion for the subsequent character</td>
</tr>
</tbody>
</table>

#### Sample Entry

The following entry, which describes the AT&T 610 terminal, is among the more complex entries in the `terminfo` file as of this writing.

```
610|610bct|ATT610|att610|AT&T610;80column;98key keyboard
am, eslok, hs, mir, mmsg, xenl, xon,
cols#80, it#8, lb#2, lines#24, lw#8, nlab#8, ws1#80,
accsc='\"aafgjjkkllmmoonopqqrresttuuvvwxz\{"|\}~\,",
bel='^G', blink='\E[5m, bold='\E[1m, cht='\E[2, cvis='\E[7251, clear='\E[H\E[J, cnorm='\E[725h\E[?21,
cr='\r, csr='\E[Hi\E[\d;\p2\dr, cub='\E[\p1%dD, cub1='\b, cud='\E[\p1dB, cud1='\E[B, cuf='\E[\p1%dC, cufl='\E[C,
```

**File Formats** 495
The sample entry shows the formats for the three types of terminfo capabilities listed: Boolean, numeric, and string. All capabilities specified in the terminfo source file must be followed by commas, including the last capability in the source file. In terminfo source files, capabilities are referenced by their capability names (as shown in the previous tables).

Boolean capabilities are specified simply by their comma separated cap names.

Numeric capabilities are followed by the character ‘#’ and then a positive integer value. Thus, in the sample, cols (which shows the number of columns available on a device) is assigned the value 80 for the AT&T 610. (Values for numeric capabilities may be specified in decimal, octal, or hexadecimal, using normal C programming language conventions.)

Finally, string-valued capabilities such as el (clear to end of line sequence) are listed by a two- to five-character capname, an ‘=’, and a string ended by the next occurrence of a comma. A delay in milliseconds may appear anywhere in such a capability, preceded by $ and enclosed in angle brackets, as in el=$<3>. Padding characters are supplied by tput. The delay can be any of the following: a number, a number followed by an asterisk, such as 5*, a number followed by a slash, such as 5/0, or a number followed by both, such as 5/*/. A ‘*’ shows that the padding required is proportional to the number of lines affected by the operation, and the amount given is the per-affected-unit padding required. (In the case of insert characters, the factor is still the number of lines affected. This is always 1 unless the device has in and the software uses it.) When a ‘*’ is specified, it is sometimes useful to give a delay of the form 3.5 to specify a delay per unit to tenths of milliseconds. (Only one decimal place is allowed.)
A ‘/’ indicates that the padding is mandatory. If a device has xon defined, the padding information is advisory and will only be used for cost estimates or when the device is in raw mode. Mandatory padding will be transmitted regardless of the setting of xon. If padding (whether advisory or mandatory) is specified for bel or flash, however, it will always be used, regardless of whether xon is specified.

terminfo offers notation for encoding special characters. Both \E and \e map to an ESCAPE character, ^x maps to a control x for any appropriate x, and the sequences \n, \l, \r, \t, \b, \f, and \s give a newline, linefeed, return, tab, backspace, formfeed, and space, respectively. Other escapes include: \^ for caret (^); \\ for backslash (\); \ for comma (,); \: for colon (:); and \0 for null. (\0 will actually produce \200, which does not terminate a string but behaves as a null character on most devices, providing CS7 is specified. (See stty(1)). Finally, characters may be given as three octal digits after a backslash (for example, \123).

Sometimes individual capabilities must be commented out. To do this, put a period before the capability name. For example, see the second ind in the example above. Note that capabilities are defined in a left-to-right order and, therefore, a prior definition will override a later definition.

Preparation of device descriptions is by imitating the description of a similar device in terminfo and building up a description gradually, using partial descriptions with vi to check that they are correct. Be aware that a very unusual device may expose deficiencies in the ability of the terminfo file to describe it or the inability of vi to work with that device. To test a new device description, set the environment variable TERMINO to the pathname of a directory containing the compiled description you are working on and programs will look there rather than in /usr/share/lib/terminfo. To get the padding for insert-line correct (if the device manufacturer did not document it) a severe test is to comment out xon, edit a large file at 9600 baud with vi, delete 16 or so lines from the middle of the screen, and then press the u key several times quickly. If the display is corrupted, more padding is usually needed. A similar test can be used for insert-character.

The number of columns on each line for the device is given by the cols numeric capability. If the device has a screen, then the number of lines on the screen is given by the lines capability. If the device wraps around to the beginning of the next line when it reaches the right margin, then it should have the am capability. If the terminal can clear its screen, leaving the cursor in the home position, then this is given by the clear string capability. If the terminal overstrikes (rather than clearing a position when a character is struck over) then it should have the os capability. If the device is a printing terminal, with no soft copy unit, specify both hc and os. If there is a way to move the cursor to the left edge of the current row, specify this as cz. (Normally this will be carriage return, control M.) If there is a way to produce an audible signal (such as a bell or a beep), specify it as bel. If, like most devices, the device uses the xon-xoff flow-control protocol, specify xon.
If there is a way to move the cursor one position to the left (such as backspace), that capability should be given as \texttt{cub1}. Similarly, sequences to move to the right, up, and down should be given as \texttt{cuf1}, \texttt{cuu1}, and \texttt{cud1}, respectively. These local cursor motions must not alter the text they pass over; for example, you would not normally use \texttt{"cuf1=\s"} because the space would erase the character moved over.

A very important point here is that the local cursor motions encoded in \texttt{terminfo} are undefined at the left and top edges of a screen terminal. Programs should never attempt to backspace around the left edge, unless \texttt{bw} is specified, and should never attempt to go up locally off the top. To scroll text up, a program goes to the bottom left corner of the screen and sends the \texttt{ind} (index) string.

To scroll text down, a program goes to the top left corner of the screen and sends the \texttt{ri} (reverse index) string. The strings \texttt{ind} and \texttt{ri} are undefined when not on their respective corners of the screen.

Parameterized versions of the scrolling sequences are \texttt{indn} and \texttt{rin}. These versions have the same semantics as \texttt{ind} and \texttt{ri}, except that they take one parameter and scroll the number of lines specified by that parameter. They are also undefined except at the appropriate edge of the screen.

The \texttt{am} capability tells whether the cursor sticks at the right edge of the screen when text is output, but this does not necessarily apply to a \texttt{cuf1} from the last column. Backward motion from the left edge of the screen is possible only when \texttt{bw} is specified. In this case, \texttt{cub1} will move to the right edge of the previous row. If \texttt{bw} is not given, the effect is undefined. This is useful for drawing a box around the edge of the screen, for example. If the device has switch selectable automatic margins, \texttt{am} should be specified in the \texttt{terminfo} source file. In this case, initialization strings should turn on this option, if possible. If the device has a command that moves to the first column of the next line, that command can be given as \texttt{nel} (newline). It does not matter if the command clears the remainder of the current line, so if the device has no \texttt{cr} and \texttt{lf} it may still be possible to craft a working \texttt{nel} out of one or both of them.

These capabilities suffice to describe hardcopy and screen terminals. Thus the AT&T 5320 hardcopy terminal is described as follows:

\begin{verbatim}
 5320|att5320|AT&T 5320 hardcopy terminal,  
am, hc, os,  
cols#132,  
bel=^G, cr=\r, cub1=\b, cud1=\n,  
dch1=\E[P, dl1=\E[M,  
ind=\n,
\end{verbatim}

while the Lear Siegler ADM–3 is described as

\begin{verbatim}
adm3 | lsi adm3,  
am, bel=^G, clear=^Z, cols#80, cr=^M, cub1=^H,  
cud1=^J, ind=^J, lines#24,
\end{verbatim}

Cursor addressing and other strings requiring parameters are described by a parameterized string capability, with \texttt{printf}-like escapes (\texttt{\%x}) in it. For example, to address the cursor, the \texttt{cup} capability is given, using two parameters: the row and...
column to address to. (Rows and columns are numbered from zero and refer to the physical screen visible to the user, not to any unseen memory.) If the terminal has memory relative cursor addressing, that can be indicated by `mrcup`.

The parameter mechanism uses a stack and special `%` codes to manipulate the stack in the manner of Reverse Polish Notation (postfix). Typically a sequence will push one of the parameters onto the stack and then print it in some format. Often more complex operations are necessary. Operations are in postfix form with the operands in the usual order. That is, to subtract 5 from the first parameter, one would use `%p1%{5}%−`.

The `%` encodings have the following meanings:

```
%:
  outputs ‘%’

% [ [: ]flags][width[.precision]][doxXs]
  as in printf, flags are [−+#] and space

%c
  print pop gives %c

%p [1-9]
  push i
th parm

%P[a-z]
  set dynamic variable [a-z] to pop

%g[a-z]
  get dynamic variable [a-z] and push it

%P[A-Z]
  set static variable [a-z] to pop

%g[A-Z]
  get static variable [a-z] and push it

%c
  push char constant c

%{nn}
  push decimal constant nn

%l
  push strlen(pop)

%+ %− %* %/ %m
  arithmetic (%m is mod): push(pop integer2 op pop integer1)

%& %| %^ %*
  bit operations: push(pop integer2 op pop integer1)

%>= %< %=
  logical operations: push(pop integer2 op pop integer1)
```
logical operations: and, or

unary operations: push(op pop)

(for ANSI terminals) add 1 to first parm, if one parm present, or first two parms, if more than one parm present

if-then-else, %e elsepart is optional; else-if's are possible ala Algol 68: %? c_1 %t b_1 %e c_2 %t b_2 %e c_3 %t b_3 %e c_4 %t b_4 %e b_5 %e. c_i are conditions, b_i are bodies.

If the "~" flag is used with "%t[doXs]", then a colon (:) must be placed between the "~" and the "~" to differentiate the flag from the binary "~" operator, for example "~16.16s".

Consider the Hewlett-Packard 2645, which, to get to row 3 and column 12, needs to be sent %a12c03Y padded for 6 milliseconds. Note that the order of the rows and columns is inverted here, and that the row and column are zero-padded as two digits. Thus its cup capability is: cup=\%a\%p2%2.2dc\%p1%2.2dY$<6>

The Micro-Term ACT-IV needs the current row and column sent preceded by a ^T, with the row and column simply encoded in binary, “cup=^T%p1%c%p2%c”. Devices that use “%c” need to be able to backspace the cursor (cub1), and to move the cursor up one line on the screen (cuu1). This is necessary because it is not always safe to transmit \n, ^D, and \r, as the system may change or discard them. (The library routines dealing with terminfo set tty modes so that tabs are never expanded, so \t is safe to send. This turns out to be essential for the Ann Arbor 4080.)

A final example is the LSI ADM-3a, which uses row and column offset by a blank character, thus “cup=\E%+%p1\%s\%+%c%p2\%s\%+%c”. After sending “\E=”, this pushes the first parameter, pushes the ASCII value for a space (32), adds them (pushing the sum on the stack in place of the two previous values), and outputs that value as a character. Then the same is done for the second parameter. More complex arithmetic is possible using the stack.

If the terminal has a fast way to home the cursor (to very upper left corner of screen) then this can be given as home; similarly a fast way of getting to the lower left-hand corner can be given as l1; this may involve going up with cuu1 from the home position, but a program should never do this itself (unless l1 does) because it can make no assumption about the effect of moving up from the home position. Note that the home position is the same as addressing to (0,0): to the top left corner of the screen, not of memory. (Thus, the \Eh sequence on Hewlett-Packard terminals cannot be used for home without losing some of the other features on the terminal.)

If the device has row or column absolute-cursor addressing, these can be given as single parameter capabilities hpa (horizontal position absolute) and vpa (vertical position absolute). Sometimes these are shorter than the more general two-parameter sequence (as with the Hewlett-Packard 2645) and can be used in preference to cup. If
there are parameterized local motions (for example, move \( n \) spaces to the right) these can be given as \( \text{cud}, \text{cub}, \text{cuf}, \text{cuu} \) with a single parameter indicating how many spaces to move. These are primarily useful if the device does not have \( \text{cup} \), such as the Tektronix 4025.

If the device needs to be in a special mode when running a program that uses these capabilities, the codes to enter and exit this mode can be given as \( \text{smcup} \) and \( \text{rmcup} \). This arises, for example, from terminals, such as the Concept, with more than one page of memory. If the device has only memory relative cursor addressing and not screen relative cursor addressing, a one screen-sized window must be fixed into the device for cursor addressing to work properly. This is also used for the Tektronix 4025, where \( \text{smcup} \) sets the command character to be the one used by terminfo. If the \( \text{smcup} \) sequence will not restore the screen after an \( \text{rmcup} \) sequence is output (to the state prior to outputting \( \text{rmcup} \)), specify \( \text{nrrmc} \).

If the terminal can clear from the current position to the end of the line, leaving the cursor where it is, this should be given as \( \text{el} \). If the terminal can clear from the beginning of the line to the current position inclusive, leaving the cursor where it is, this should be given as \( \text{el1} \). If the terminal can clear from the current position to the end of the display, then this should be given as \( \text{ed} \). \( \text{ed} \) is only defined from the first column of a line. (Thus, it can be simulated by a request to delete a large number of lines, if a true \( \text{ed} \) is not available.)

If the terminal can open a new blank line before the line where the cursor is, this should be given as \( \text{il1} \); this is done only from the first position of a line. The cursor must then appear on the newly blank line. If the terminal can delete the line which the cursor is on, then this should be given as \( \text{dl1} \); this is done only from the first position on the line to be deleted. Versions of \( \text{il1} \) and \( \text{dl1} \) which take a single parameter and insert or delete that many lines can be given as \( \text{il} \) and \( \text{dl} \).

If the terminal has a settable destructive scrolling region (like the VT100) the command to set this can be described with the \( \text{csr} \) capability, which takes two parameters: the top and bottom lines of the scrolling region. The cursor position is, alas, undefined after using this command. It is possible to get the effect of insert or delete line using this command — the \( \text{sc} \) and \( \text{rc} \) (save and restore cursor) commands are also useful. Inserting lines at the top or bottom of the screen can also be done using \( \text{ri} \) or \( \text{ind} \) on many terminals without a true insert/delete line, and is often faster even on terminals with those features.

To determine whether a terminal has destructive scrolling regions or non-destructive scrolling regions, create a scrolling region in the middle of the screen, place data on the bottom line of the scrolling region, move the cursor to the top line of the scrolling region, and do a reverse index (\( \text{ri} \)) followed by a delete line (\( \text{dl1} \)) or index (\( \text{ind} \)). If the data that was originally on the bottom line of the scrolling region was restored into the scrolling region by the \( \text{dl1} \) or \( \text{ind} \), then the terminal has non-destructive scrolling regions. Otherwise, it has destructive scrolling regions. Do not specify \( \text{csr} \) if the terminal has non-destructive scrolling regions, unless \( \text{ind}, \text{ri}, \text{indn}, \text{rin}, \text{dl}, \text{and dl1} \) all simulate destructive scrolling.
If the terminal has the ability to define a window as part of memory, which all commands affect, it should be given as the parameterized string wind. The four parameters are the starting and ending lines in memory and the starting and ending columns in memory, in that order.

If the terminal can retain display memory above, then the da capability should be given; if display memory can be retained below, then db should be given. These indicate that deleting a line or scrolling a full screen may bring non-blank lines up from below or that scrolling back with ri may bring down non-blank lines.

There are two basic kinds of intelligent terminals with respect to insert/delete character operations which can be described using terminfo. The most common insert/delete character operations affect only the characters on the current line and shift characters off the end of the line rigidly. Other terminals, such as the Concept 100 and the Perkin Elmer Owl, make a distinction between typed and untyped blanks on the screen, shifting upon an insert or delete only to an untyped blank on the screen which is either eliminated, or expanded to two untyped blanks. You can determine the kind of terminal you have by clearing the screen and then typing text separated by cursor motions. Type “abc def” using local cursor motions (not spaces) between the abc and the def. Then position the cursor before the abc and put the terminal in insert mode. If typing characters causes the rest of the line to shift rigidly and characters to fall off the end, then your terminal does not distinguish between blanks and untyped positions. If the abc shifts over to the def which then move together around the end of the current line and onto the next as you insert, you have the second type of terminal, and should give the capability in, which stands for “insert null.” While these are two logically separate attributes (one line versus multiline insert mode, and special treatment of untyped spaces) we have seen no terminals whose insert mode cannot be described with the single attribute.

terminfo can describe both terminals that have an insert mode and terminals which send a simple sequence to open a blank position on the current line. Give as smir the sequence to get into insert mode. Give as rmir the sequence to leave insert mode.

Now give as ich1 any sequence needed to be sent just before sending the character to be inserted. Most terminals with a true insert mode will not give ich1; terminals that send a sequence to open a screen position should give it here. (If your terminal has both, insert mode is usually preferable to ich1. Do not give both unless the terminal actually requires both to be used in combination.) If post-insert padding is needed, give this as a number of milliseconds padding in ip (a string option). Any other sequence which may need to be sent after an insert of a single character may also be given in ip. If your terminal needs both to be placed into an ‘insert mode’ and a special code to precede each inserted character, then both smir/rmir and ich1 can be given, and both will be used. The ich capability, with one parameter, n, will insert n blanks.

If padding is necessary between characters typed while not in insert mode, give this as a number of milliseconds padding in rmp.
It is occasionally necessary to move around while in insert mode to delete characters on the same line (for example, if there is a tab after the insertion position). If your terminal allows motion while in insert mode you can give the capability \texttt{mir} to speed up inserting in this case. Omitting \texttt{mir} will affect only speed. Some terminals (notably Datamedia’s) must not have \texttt{mir} because of the way their insert mode works.

Finally, you can specify \texttt{dch1} to delete a single character, \texttt{dch} with one parameter, \texttt{n}, to delete \texttt{n} characters, and delete mode by giving \texttt{smdc} and \texttt{rmdc} to enter and exit delete mode (any mode the terminal needs to be placed in for \texttt{dch1} to work).

A command to erase \texttt{n} characters (equivalent to outputting \texttt{n} blanks without moving the cursor) can be given as \texttt{ech} with one parameter.

Your device may have one or more kinds of display attributes that allow you to highlight selected characters when they appear on the screen. The following display modes (shown with the names by which they are set) may be available: a blinking screen (\texttt{blink}), bold or extra-bright characters (\texttt{bold}), dim or half-bright characters (\texttt{dim}), blanking or invisible text (\texttt{invis}), protected text (\texttt{prot}), a reverse-video screen (\texttt{rev}), and an alternate character set (\texttt{smacs} to enter this mode and \texttt{rmacs} to exit it). (If a command is necessary before you can enter alternate character set mode, give the sequence in \texttt{enacs} or “enable alternate-character-set” mode.) Turning on any of these modes singly may or may not turn off other modes.

\texttt{sgr0} should be used to turn off all video enhancement capabilities. It should always be specified because it represents the only way to turn off some capabilities, such as \texttt{dim} or \texttt{blink}.

You should choose one display method as \texttt{standout mode} and use it to highlight error messages and other kinds of text to which you want to draw attention. Choose a form of display that provides strong contrast but that is easy on the eyes. (We recommend reverse-video plus half-bright or reverse-video alone.) The sequences to enter and exit standout mode are given as \texttt{smso} and \texttt{rmso}, respectively. If the code to change into or out of standout mode leaves one or even two blank spaces on the screen, as the TVI 912 and Teleray 1061 do, then \texttt{xmc} should be given to tell how many spaces are left.

Sequences to begin underlining and end underlining can be specified as \texttt{smul} and \texttt{rmul}, respectively. If the device has a sequence to underline the current character and to move the cursor one space to the right (such as the Micro-Term MIME), this sequence can be specified as \texttt{uc}.

Terminals with the “magic cookie” glitch (\texttt{xmc}) deposit special “cookies” when they receive mode-setting sequences, which affect the display algorithm rather than having extra bits for each character. Some terminals, such as the Hewlett-Packard 2621, automatically leave standout mode when they move to a new line or the cursor is addressed. Programs using standout mode should exit standout mode before moving the cursor or sending a newline, unless the \texttt{msgr} capability, asserting that it is safe to move in standout mode, is present.
If the terminal has a way of flashing the screen to indicate an error quietly (a bell replacement), then this can be given as \texttt{flash}; it must not move the cursor. A good flash can be done by changing the screen into reverse video, pad for 200 ms, then return the screen to normal video.

If the cursor needs to be made more visible than normal when it is not on the bottom line (to make, for example, a non-blinking underline into an easier to find block or blinking underline) give this sequence as \texttt{cvvis}. The boolean \texttt{chts} should also be given. If there is a way to make the cursor completely invisible, give that as \texttt{civis}. The capability \texttt{cnorm} should be given which undoes the effects of either of these modes.

If your terminal generates underlined characters by using the underline character (with no special sequences needed) even though it does not otherwise overstrike characters, then you should specify the capability \texttt{ul}. For devices on which a character overstriking another leaves both characters on the screen, specify the capability \texttt{os}. If overstrikes are erasable with a blank, then this should be indicated by specifying \texttt{eo}.

If there is a sequence to set arbitrary combinations of modes, this should be given as \texttt{sgr} (set attributes), taking nine parameters. Each parameter is either 0 or non-zero, as the corresponding attribute is on or off. The nine parameters are, in order: standout, underline, reverse, blink, dim, bold, blank, protect, alternate character set. Not all modes need to be supported by \texttt{sgr}; only those for which corresponding separate attribute commands exist should be supported. For example, let’s assume that the terminal in question needs the following escape sequences to turn on various modes.

<table>
<thead>
<tr>
<th>tparm parameter</th>
<th>attribute</th>
<th>escape sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>\texttt{\E[0m}</td>
<td></td>
</tr>
<tr>
<td>p1</td>
<td>standout</td>
<td>\texttt{\E[0;4;7m}</td>
</tr>
<tr>
<td>p2</td>
<td>underline</td>
<td>\texttt{\E[0;3m}</td>
</tr>
<tr>
<td>p3</td>
<td>reverse</td>
<td>\texttt{\E[0;4m}</td>
</tr>
<tr>
<td>p4</td>
<td>blink</td>
<td>\texttt{\E[0;5m}</td>
</tr>
<tr>
<td>p5</td>
<td>dim</td>
<td>\texttt{\E[0;7m}</td>
</tr>
<tr>
<td>p6</td>
<td>bold</td>
<td>\texttt{\E[0;3;4m}</td>
</tr>
<tr>
<td>p7</td>
<td>invis</td>
<td>\texttt{\E[0;8m}</td>
</tr>
<tr>
<td>p8</td>
<td>protect</td>
<td>not available</td>
</tr>
<tr>
<td>p9</td>
<td>altcharset</td>
<td>\texttt{^O (off) ^N (on)}</td>
</tr>
</tbody>
</table>
Note that each escape sequence requires a 0 to turn off other modes before turning on its own mode. Also note that, as suggested above, standout is set up to be the combination of reverse and dim. Also, because this terminal has no bold mode, bold is set up as the combination of reverse and underline. In addition, to allow combinations, such as underline+blink, the sequence to use would be \E[0;3;5m. The terminal doesn’t have protect mode, either, but that cannot be simulated in any way, so p8 is ignored. The altcharset mode is different in that it is either ^O or ^N, depending on whether it is off or on. If all modes were to be turned on, the sequence would be \E[0;3;4;5;7;8m^N.

Now look at when different sequences are output. For example, ;3 is output when either p2 or p6 is true, that is, if either underline or bold modes are turned on. Writing out the above sequences, along with their dependencies, gives the following:

<table>
<thead>
<tr>
<th>sequence</th>
<th>when to output</th>
<th>terminfo translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>\E[0</td>
<td>always</td>
<td>\E[0</td>
</tr>
<tr>
<td>;3</td>
<td>if p2 or p6</td>
<td>%?%p2%p6%1%t;3%;</td>
</tr>
<tr>
<td>;4</td>
<td>if p1 or p3 or p6</td>
<td>%?%p1%p3%1%p6%1%t;4%;</td>
</tr>
<tr>
<td>;5</td>
<td>if p4</td>
<td>%?%p4%t;5%;</td>
</tr>
<tr>
<td>;7</td>
<td>if p1 or p5</td>
<td>%?%p1%p5%1%t;7%;</td>
</tr>
<tr>
<td>;8</td>
<td>if p7</td>
<td>%?%p7%t;8%;</td>
</tr>
<tr>
<td>m</td>
<td>always</td>
<td>m</td>
</tr>
<tr>
<td>^N or ^O</td>
<td>if p9 ^N, else ^O</td>
<td>%?%p9%t^N%e^O%;</td>
</tr>
</tbody>
</table>

Putting this all together into the sgr sequence gives:

sgr=\E[0%?%p2%p6%1%t;3%;%?%p1%p3%1%p6%1%t;4%;%?%p4%t;5%;%?%p1%p5%1%t;7%;%?%p7%t;8%;m%?%p9%t^N%e^O%;

Remember that sgr and sgr0 must always be specified.

If the device has a keypad that transmits sequences when the keys are pressed, this information can also be specified. Note that it is not possible to handle devices where the keypad only works in local (this applies, for example, to the unshifted Hewlett-Packard 2621 keys). If the keypad can be set to transmit or not transmit, specify these sequences as smkx and rmkx. Otherwise the keypad is assumed to always transmit.

The sequences sent by the left arrow, right arrow, up arrow, down arrow, and home keys can be given as kcub1, kcufl, kcuu1, kcul, and khome, respectively. If there are function keys such as f0, f1, ..., f63, the sequences they send can be specified as kf0, kf1, ..., kf63. If the first 11 keys have labels other than the default f0 through f10, the labels can be given as lf0, lf1, ..., lf10. The codes

---

**Section 1-8: Keypad**

<table>
<thead>
<tr>
<th>terminfo(4) Section 1-8: Keypad</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>9</strong></td>
</tr>
<tr>
<td><strong>505</strong></td>
</tr>
</tbody>
</table>
transmitted by certain other special keys can be given: kll (home down), kbs (backspace), ktb (clear all tabs), kct (clear the tab stop in this column), kcl (clear screen or erase key), kdch1 (delete character), kdll (delete line), krmir (exit insert mode), kel (clear to end of line), ked (clear to end of screen), kich1 (insert character or enter insert mode), kill (insert line), knp (next page), kpp (previous page), kind (scroll forward/down), kri (scroll backward/up), khts (set a tab stop in this column). In addition, if the keypad has a 3 by 3 array of keys including the four arrow keys, the other five keys can be given as ka1, ka3, kb2, kc1, and kc3. These keys are useful when the effects of a directional pad are needed. Further keys are defined above in the capabilities list.

Strings to program function keys can be specified as pfkey, pfloc, and pfx. A string to program screen labels should be specified as pln. Each of these strings takes two parameters: a function key identifier and a string to program it with. pfkey causes pressing the given key to be the same as the user typing the given string; pfloc causes the string to be executed by the terminal in local mode; and pfx causes the string to be transmitted to the computer. The capabilities nlab, lw and lh define the number of programmable screen labels and their width and height. If there are commands to turn the labels on and off, give them in smln and rmln. smln is normally output after one or more pln sequences to make sure that the change becomes visible.

If the device has hardware tabs, the command to advance to the next tab stop can be given as ht (usually control I). A "backtab" command that moves leftward to the next tab stop can be given as cbt. By convention, if tty modes show that tabs are being expanded by the computer rather than being sent to the device, programs should not use ht or cbt (even if they are present) because the user may not have the tab stops properly set. If the device has hardware tabs that are initially set every n spaces when the device is powered up, the numeric parameter it is given, showing the number of spaces the tabs are set to. This is normally used by tput init (see tput(1)) to determine whether to set the mode for hardware tab expansion and whether to set the tab stops. If the device has tab stops that can be saved in nonvolatile memory, the terminfo description can assume that they are properly set. If there are commands to set and clear tab stops, they can be given as tbc (clear all tab stops) and hts (set a tab stop in the current column of every row).

Other capabilities include: is1, is2, and is3, initialization strings for the device; iprog, the path name of a program to be run to initialize the device; and if, the name of a file containing long initialization strings. These strings are expected to set the device into modes consistent with the rest of the terminfo description. They must be sent to the device each time the user logs in and be output in the following order: run the program iprog; output is1; output is2; set the margins using mgc, smgl and smgr; set the tabs using tbc and hts; print the file if; and finally output is3. This is usually done using the init option of tput.

Most initialization is done with is2. Special device modes can be set up without duplicating strings by putting the common sequences in is2 and special cases in is1 and is3. Sequences that do a reset from a totally unknown state can be given as rs1, rs2, rf, and rs3, analogous to is1, is2, is3, and if. (The method using files, if
and rf, is used for a few terminals, from /usr/share/lib/tabset/*; however, the recommended method is to use the initialization and reset strings.) These strings are output by tput reset, which is used when the terminal gets into a wedged state. Commands are normally placed in rs1, rs2, rs3, and rf only if they produce annoying effects on the screen and are not necessary when logging in. For example, the command to set a terminal into 80-column mode would normally be part of is2, but on some terminals it causes an annoying glitch on the screen and is not normally needed because the terminal is usually already in 80-column mode.

If a more complex sequence is needed to set the tabs than can be described by using tbc and hts, the sequence can be placed in is2 or if.

Any margin can be cleared with mgc. (For instructions on how to specify commands to set and clear margins, see "Margins" below under "PRINTER CAPABILITIES").

**Section 1-10: Delays**

Certain capabilities control padding in the tty driver. These are primarily needed by hard-copy terminals, and are used by tput init to set tty modes appropriately. Delays embedded in the capabilities cr, ind, cub1, ff, and tab can be used to set the appropriate delay bits to be set in the tty driver. If pb (padding baud rate) is given, these values can be ignored at baud rates below the value of pb.

**Section 1-11: Status Lines**

If the terminal has an extra “status line” that is not normally used by software, this fact can be indicated. If the status line is viewed as an extra line below the bottom line, into which one can cursor address normally (such as the Heathkit h19’s 25th line, or the 24th line of a VT100 which is set to a 23-line scrolling region), the capability hs should be given. Special strings that go to a given column of the status line and return from the status line can be given as tsl and fsl. (fsl must leave the cursor position in the same place it was before tsl. If necessary, the sc and rc strings can be included in tsl and fsl to get this effect.) The capability tsl takes one parameter, which is the column number of the status line the cursor is to be moved to.

If escape sequences and other special commands, such as tab, work while in the status line, the flag eslok can be given. A string which turns off the status line (or otherwise erases its contents) should be given as dsl. If the terminal has commands to save and restore the position of the cursor, give them as sc and rc. The status line is normally assumed to be the same width as the rest of the screen, for example, cols. If the status line is a different width (possibly because the terminal does not allow an entire line to be loaded) the width, in columns, can be indicated with the numeric parameter wsl.

**Section 1-12: Line Graphics**

If the device has a line drawing alternate character set, the mapping of glyph to character would be given in acsc. The definition of this string is based on the alternate character set used in the DEC VT100 terminal, extended slightly with some characters from the AT&T 4410v1 terminal.

<table>
<thead>
<tr>
<th>Glyph Name</th>
<th>vt100+ Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrow pointing right</td>
<td>+</td>
</tr>
</tbody>
</table>
The best way to describe a new device's line graphics set is to add a third column to the above table with the characters for the new device that produce the appropriate glyph when the device is in the alternate character set mode. For example,
Now write down the characters left to right, as in "acsc=lRmFkTjGq\,x."

In addition, terminfo allows you to define multiple character sets. See Section 2-5 for details.

Let us define two methods of color manipulation: the Tektronix method and the HP method. The Tektronix method uses a set of N predefined colors (usually 8) from which a user can select "current" foreground and background colors. Thus a terminal can support up to N colors mixed into N*N color-pairs to be displayed on the screen at the same time. When using an HP method the user cannot define the foreground independently of the background, or vice-versa. Instead, the user must define an entire color-pair at once. Up to M color-pairs, made from 2*M different colors, can be defined this way. Most existing color terminals belong to one of these two classes of terminals.

The numeric variables colors and pairs define the number of colors and color-pairs that can be displayed on the screen at the same time. If a terminal can change the definition of a color (for example, the Tektronix 4100 and 4200 series terminals), this should be specified with ccc (can change color). To change the definition of a color (Tektronix 4200 method), use initc (initialize color). It requires four arguments: color number (ranging from 0 to colors - 1) and three RGB (red, green, and blue) values or three HLS colors (Hue, Lightness, Saturation). Ranges of RGB and HLS values are terminal dependent.

Tektronix 4100 series terminals only use HLS color notation. For such terminals (or dual-mode terminals to be operated in HLS mode) one must define a boolean variable hls; that would instruct the curses init_color routine to convert its RGB arguments to HLS before sending them to the terminal. The last three arguments to the initc string would then be HLS values.

If a terminal can change the definitions of colors, but uses a color notation different from RGB and HLS, a mapping to either RGB or HLS must be developed.

To set current foreground or background to a given color, use setaf (set ANSI foreground) and setab (set ANSI background). They require one parameter: the number of the color. To initialize a color-pair (HP method), use initp (initialize pair). It requires seven parameters: the number of a color-pair (range=0 to pairs-1), and six...
RGB values: three for the foreground followed by three for the background. (Each of these groups of three should be in the order RGB.) When initc or initp are used, RGB or HLS arguments should be in the order "red, green, blue" or "hue, lightness, saturation"), respectively. To make a color-pair current, use scp (set color-pair). It takes one parameter, the number of a color-pair.

Some terminals (for example, most color terminal emulators for PCs) erase areas of the screen with current background color. In such cases, bce (background color erase) should be defined. The variable op (original pair) contains a sequence for setting the foreground and the background colors to what they were at the terminal start-up time. Similarly, oc (original colors) contains a control sequence for setting all colors (for the Tektronix method) or color-pairs (for the HP method) to the values they had at the terminal start-up time.

Some color terminals substitute color for video attributes. Such video attributes should not be combined with colors. Information about these video attributes should be packed into the ncv (no color video) variable. There is a one-to-one correspondence between the nine least significant bits of that variable and the video attributes. The following table depicts this correspondence.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Bit Position</th>
<th>Decimal Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A_STANDOUT</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>A_UNDERLINE</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>A_REVERSE</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>A_BLINK</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>A_DIM</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>A_BOLD</td>
<td>5</td>
<td>32</td>
</tr>
<tr>
<td>A_INVIS</td>
<td>6</td>
<td>64</td>
</tr>
<tr>
<td>A_PROTECT</td>
<td>7</td>
<td>128</td>
</tr>
<tr>
<td>A_ALTCHARSET</td>
<td>8</td>
<td>256</td>
</tr>
</tbody>
</table>

When a particular video attribute should not be used with colors, the corresponding ncv bit should be set to 1; otherwise it should be set to zero. To determine the information to pack into the ncv variable, you must add together the decimal values corresponding to those attributes that cannot coexist with colors. For example, if the terminal uses colors to simulate reverse video (bit number 2 and decimal value 4) and bold (bit number 5 and decimal value 32), the resulting value for ncv will be 36 (4 + 32).

If the terminal requires other than a null (zero) character as a pad, then this can be given as pad. Only the first character of the pad string is used. If the terminal does not have a pad character, specify npc.
If the terminal can move up or down half a line, this can be indicated with **hu** (half-line up) and **hd** (half-line down). This is primarily useful for superscripts and subscripts on hardcopy terminals. If a hardcopy terminal can eject to the next page (**form feed**), give this as **ff** (usually control L).

If there is a command to repeat a given character a given number of times (to save time transmitting a large number of identical characters) this can be indicated with the parameterized string **rep**. The first parameter is the character to be repeated and the second is the number of times to repeat it. Thus, \texttt{tparm(repeat_char, 'x', 10)} is the same as xxxxxxxxxxxx.

If the terminal has a settable command character, such as the Tektronix 4025, this can be indicated with **cmdch**. A prototype command character is chosen which is used in all capabilities. This character is given in the **cmdch** capability to identify it. The following convention is supported on some systems: If the environment variable **CC** exists, all occurrences of the prototype character are replaced with the character in **CC**.

Terminal descriptions that do not represent a specific kind of known terminal, such as **switch**, **dialup**, **patch**, and **network**, should include the **gn** (generic) capability so that programs can complain that they do not know how to talk to the terminal. (This capability does not apply to virtual terminal descriptions for which the escape sequences are known.) If the terminal is one of those supported by the system virtual terminal protocol, the terminal number can be given as **vt**. A line-turn-around sequence to be transmitted before doing reads should be specified in **rfi**.

If the device uses xon/xoff handshaking for flow control, give **xon**. Padding information should still be included so that routines can make better decisions about costs, but actual pad characters will not be transmitted. Sequences to turn on and off xon/xoff handshaking may be given in **smxon** and **rmxon**. If the characters used for handshaking are not ^S and ^Q, they may be specified with **xonc** and **xoffc**.

If the terminal has a “meta key” which acts as a shift key, setting the 8th bit of any character transmitted, this fact can be indicated with **km**. Otherwise, software will assume that the 8th bit is parity and it will usually be cleared. If strings exist to turn this “meta mode” on and off, they can be given as **smm** and **rmm**.

If the terminal has more lines of memory than will fit on the screen at once, the number of lines of memory can be indicated with **lm**. A value of **lm**#0 indicates that the number of lines is not fixed, but that there is still more memory than fits on the screen.

Media copy strings which control an auxiliary printer connected to the terminal can be given as **mc0**: print the contents of the screen, **mc4**: turn off the printer, and **mc5**: turn on the printer. When the printer is on, all text sent to the terminal will be sent to the printer. A variation, **mc5p**, takes one parameter, and leaves the printer on for as many characters as the value of the parameter, then turns the printer off. The parameter should not exceed 255. If the text is not displayed on the terminal screen when the printer is on, specify **mc5i** (silent printer). All text, including **mc4**, is transparently passed to the printer while an **mc5p** is in effect.
The working model used by `terminfo` fits most terminals reasonably well. However, some terminals do not completely match that model, requiring special support by `terminfo`. These are not meant to be construed as deficiencies in the terminals; they are just differences between the working model and the actual hardware. They may be unusual devices or, for some reason, do not have all the features of the `terminfo` model implemented.

Terminals that cannot display tilde (-) characters, such as certain Hazeltine terminals, should indicate `hz`.

Terminals that ignore a linefeed immediately after an `am` wrap, such as the Concept 100, should indicate `xenl`. Those terminals whose cursor remains on the right-most column until another character has been received, rather than wrapping immediately upon receiving the right-most character, such as the VT100, should also indicate `xenl`.

If `el` is required to get rid of standout (instead of writing normal text on top of it), `xhp` should be given.

Those Teleray terminals whose tabs turn all characters moved over to blanks, should indicate `xt` (destructive tabs). This capability is also taken to mean that it is not possible to position the cursor on top of a “magic cookie.” Therefore, to erase standout mode, it is necessary, instead, to use delete and insert line.

Those Beehive Superbee terminals which do not transmit the escape or control-`C` characters, should specify `xsb`, indicating that the f1 key is to be used for escape and the f2 key for control C.

If there are two very similar terminals, one can be defined as being just like the other with certain exceptions. The string capability `use` can be given with the name of the similar terminal. The capabilities given before `use` override those in the terminal type invoked by `use`. A capability can be canceled by placing `xx` to the left of the capability definition, where `xx` is the capability. For example, the entry

```
att4424-2|Teletype4424 in display function group ii,
rev@, sgr@, smul@, use=att4424,
```

defines an AT&T4424 terminal that does not have the `rev`, `sgr`, and `smul` capabilities, and hence cannot do highlighting. This is useful for different modes for a terminal, or for different user preferences. More than one `use` capability may be given.

The `terminfo` database allows you to define capabilities of printers as well as terminals. To find out what capabilities are available for printers as well as for terminals, see the two lists under "DEVICE CAPABILITIES" that list capabilities by variable and by capability name.

Because parameterized string capabilities work only with integer values, we recommend that `terminfo` designers create strings that expect numeric values that have been rounded. Application designers should note this and should always round values to the nearest integer before using them with a parameterized string capability.
A printer’s resolution is defined to be the smallest spacing of characters it can achieve. In general printers have independent resolution horizontally and vertically. Thus the vertical resolution of a printer can be determined by measuring the smallest achievable distance between consecutive printing baselines, while the horizontal resolution can be determined by measuring the smallest achievable distance between the left-most edges of consecutive printed, identical, characters.

All printers are assumed to be capable of printing with a uniform horizontal and vertical resolution. The view of printing that `termcap` currently presents is one of printing inside a uniform matrix: All characters are printed at fixed positions relative to each “cell” in the matrix; furthermore, each cell has the same size given by the smallest horizontal and vertical step sizes dictated by the resolution. (The cell size can be changed as will be seen later.)

Many printers are capable of “proportional printing,” where the horizontal spacing depends on the size of the character last printed. `termcap` does not make use of this capability, although it does provide enough capability definitions to allow an application to simulate proportional printing.

A printer must not only be able to print characters as close together as the horizontal and vertical resolutions suggest, but also of “moving” to a position an integral multiple of the smallest distance away from a previous position. Thus printed characters can be spaced apart a distance that is an integral multiple of the smallest distance, up to the length or width of a single page.

Some printers can have different resolutions depending on different “modes.” In “normal mode,” the existing `termcap` capabilities are assumed to work on columns and lines, just like a video terminal. Thus the old `lines` capability would give the length of a page in lines, and the `cols` capability would give the width of a page in columns. In “micro mode,” many `termcap` capabilities work on increments of lines and columns. With some printers the micro mode may be concomitant with normal mode, so that all the capabilities work at the same time.

The printing resolution of a printer is given in several ways. Each specifies the resolution as the number of smallest steps per distance:

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Number of Smallest Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>orhi</td>
<td>Steps per inch horizontally</td>
</tr>
<tr>
<td>orvi</td>
<td>Steps per inch vertically</td>
</tr>
<tr>
<td>orc</td>
<td>Steps per column</td>
</tr>
<tr>
<td>orl</td>
<td>Steps per line</td>
</tr>
</tbody>
</table>

When printing in normal mode, each character printed causes movement to the next column, except in special cases described later; the distance moved is the same as the per-column resolution. Some printers cause an automatic movement to the next line.
when a character is printed in the rightmost position; the distance moved vertically is
the same as the per-line resolution. When printing in micro mode, these distances can
be different, and may be zero for some printers.

Specification of Printer Resolution
Automatic Motion after Printing

Normal Mode:

orc  Steps moved horizontally
orl  Steps moved vertically

Micro Mode:

mcs  Steps moved horizontally
mls  Steps moved vertically

Some printers are capable of printing wide characters. The distance moved when a
wide character is printed in normal mode may be different from when a regular width
character is printed. The distance moved when a wide character is printed in micro
mode may also be different from when a regular character is printed in micro mode,
but the differences are assumed to be related: If the distance moved for a regular
character is the same whether in normal mode or micro mode (mcs=orc), then the
distance moved for a wide character is also the same whether in normal mode or
micro mode. This doesn’t mean the normal character distance is necessarily the same
as the wide character distance, just that the distances don’t change with a change in
normal to micro mode. However, if the distance moved for a regular character is
different in micro mode from the distance moved in normal mode (mcs<orc), the
micro mode distance is assumed to be the same for a wide character printed in micro
mode, as the table below shows.

Specification of Printer Resolution
Automatic Motion after Printing Wide Character

Normal Mode or Micro Mode (mcs = orc):
sp
widcs  Steps moved horizontally

Micro Mode (mcs < orc):

mcs  Steps moved horizontally

There may be control sequences to change the number of columns per inch (the
character pitch) and to change the number of lines per inch (the line pitch). If these are
used, the resolution of the printer changes, but the type of change depends on the
printer.
Specification of Printer Resolution
Changing the Character/Line Pitches

cpi Change character pitch
cpix If set, cpi changes orhi, otherwise changes
orc
lpi Change line pitch
lpix If set, lpi changes orvi, otherwise changes
orl
chr Change steps per column
cvr Change steps per line

The cpi and lpi string capabilities are each used with a single argument, the pitch in columns (or characters) and lines per inch, respectively. The chr and cvr string capabilities are each used with a single argument, the number of steps per column and line, respectively.

Using any of the control sequences in these strings will imply a change in some of the values of orc, orhi, orl, and orvi. Also, the distance moved when a wide character is printed, widcs, changes in relation to orc. The distance moved when a character is printed in micro mode, mcs, changes similarly, with one exception: if the distance is 0 or 1, then no change is assumed (see items marked with * in the following table).

Programs that use cpi, lpi, chr, or cvr should recalculate the printer resolution (and should recalculate other values—see "Effect of Changing Printing Resolution" under "Dot-Mapped Graphics").

<table>
<thead>
<tr>
<th>Specification of Printer Resolution</th>
<th>Effects of Changing the Character/Line Pitches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before</td>
<td>After</td>
</tr>
<tr>
<td>Using cpi with cpix clear:</td>
<td></td>
</tr>
<tr>
<td>$bold orhi $ orhi</td>
<td></td>
</tr>
<tr>
<td>$bold orc $ $bold orc = bold orhi over V sub italic cpi$</td>
<td></td>
</tr>
<tr>
<td>Using cpi with cpix set:</td>
<td></td>
</tr>
<tr>
<td>$bold orhi $ $bold orhi = bold orc cdot V sub italic cpi$</td>
<td></td>
</tr>
<tr>
<td>$bold orc $ $bold orc$</td>
<td></td>
</tr>
<tr>
<td>Using lpi with lpix clear:</td>
<td></td>
</tr>
<tr>
<td>$bold orvi $ $bold orvi$</td>
<td></td>
</tr>
<tr>
<td>$bold orl $ $bold orl = bold orvi over V sub italic lpi$</td>
<td></td>
</tr>
<tr>
<td>Using lpi with lpix set:</td>
<td></td>
</tr>
<tr>
<td>$bold orvi $ $bold orvi = bold orl cdot V sub italic lpi$</td>
<td></td>
</tr>
<tr>
<td>$bold orl $ $bold orl$</td>
<td></td>
</tr>
<tr>
<td>Using chr:</td>
<td></td>
</tr>
</tbody>
</table>
Section 2-4: Capabilities that Cause Movement

In the following descriptions, “movement” refers to the motion of the “current position.” With video terminals this would be the cursor; with some printers this is the carriage position. Other printers have different equivalents. In general, the current position is where a character would be displayed if printed.

In these descriptions, \texttt{V sub italic cpi}, \texttt{V sub italic lpi}, \texttt{V sub italic chr}, and \texttt{V sub italic cvr} are the arguments used with \texttt{cpi}, \texttt{lpi}, \texttt{chr}, and \texttt{cvr}, respectively. The prime marks (‘) indicate the old values.

In the following descriptions, “movement” refers to the motion of the “current position.” With video terminals this would be the cursor; with some printers this is the carriage position. Other printers have different equivalents. In general, the current position is where a character would be displayed if printed.

\texttt{terminfo} has string capabilities for control sequences that cause movement a number of full columns or lines. It also has equivalent string capabilities for control sequences that cause movement a number of smallest steps.

String Capabilities for Motion

- \texttt{mcub1} Move 1 step left
- \texttt{mcuf1} Move 1 step right
- \texttt{mcu1} Move 1 step up
- \texttt{mcud1} Move 1 step down
- \texttt{mcub} Move \(N\) steps left
- \texttt{mcuf} Move \(N\) steps right
- \texttt{mcuu} Move \(N\) steps up
- \texttt{mcud} Move \(N\) steps down
- \texttt{mhpa} Move \(N\) steps from the left
- \texttt{mvpa} Move \(N\) steps from the top

The latter six strings are each used with a single argument, \(N\).

Sometimes the motion is limited to less than the width or length of a page. Also, some printers don’t accept absolute motion to the left of the current position. \texttt{terminfo} has capabilities for specifying these limits.

Limits to Motion

- \texttt{mjump} Limit on use of \texttt{mcub1}, \texttt{mcuf1}, \texttt{mcu1}, \texttt{mcud1}
- \texttt{maddr} Limit on use of \texttt{mhpa}, \texttt{mvpa}
- \texttt{xhpa} If set, \texttt{hpa} and \texttt{mhpa} can’t move left
- \texttt{xvpa} If set, \texttt{vpa} and \texttt{mvpa} can’t move up
If a printer needs to be in a “micro mode” for the motion capabilities described above to work, there are string capabilities defined to contain the control sequence to enter and exit this mode. A boolean is available for those printers where using a carriage return causes an automatic return to normal mode.

Entering/Exiting Micro Mode

smicm  Enter micro mode
rmicm  Exit micro mode
crm    Using cr exits micro mode

The movement made when a character is printed in the rightmost position varies among printers. Some make no movement, some move to the beginning of the next line, others move to the beginning of the same line. terminfo has boolean capabilities for describing all three cases.

What Happens After Character Printed in Rightmost Position

sam    Automatic move to beginning of same line

Some printers can be put in a mode where the normal direction of motion is reversed. This mode can be especially useful when there are no capabilities for leftward or upward motion, because those capabilities can be built from the motion reversal capability and the rightward or downward motion capabilities. It is best to leave it up to an application to build the leftward or upward capabilities, though, and not enter them in the terminfo database. This allows several reverse motions to be strung together without intervening wasted steps that leave and reenter reverse mode.

Entering/Exiting Reverse Modes

slm    Reverse sense of horizontal motions
rlm    Restore sense of horizontal motions
sum    Reverse sense of vertical motions
rum    Restore sense of vertical motions

While sense of horizontal motions reversed:
mcub1  Move 1 step right
mcuf1  Move 1 step left
mcub   Move N steps right
mcuf   Move N steps left
cub1   Move 1 column right
cuf1   Move 1 column left
cub    Move N columns right
cuf    Move N columns left

While sense of vertical motions reversed:
mcuu1  Move 1 step down
mcud1  Move 1 step up
mcuu  Move N steps down
mccd Move N steps up
cuu1 Move 1 line down
cud1 Move 1 line up
cuu  Move N lines down
cud  Move N lines up

The reverse motion modes should not affect the mvpa and mhpa absolute motion capabilities. The reverse vertical motion mode should, however, also reverse the action of the line “wrapping” that occurs when a character is printed in the right-most position. Thus printers that have the standard terminfo capability am defined should experience motion to the beginning of the previous line when a character is printed in the right-most position under reverse vertical motion mode.

The action when any other motion capabilities are used in reverse motion modes is not defined; thus, programs must exit reverse motion modes before using other motion capabilities.

Two miscellaneous capabilities complete the list of new motion capabilities. One of these is needed for printers that move the current position to the beginning of a line when certain control characters, such as “line-feed” or “form-feed,” are used. The other is used for the capability of suspending the motion that normally occurs after printing a character.

Miscellaneous Motion Strings

doctor List of control characters causing cr
zerom Prevent auto motion after printing next single character

Margins

terminfo provides two strings for setting margins on terminals: one for the left and one for the right margin. Printers, however, have two additional margins, for the top and bottom margins of each page. Furthermore, some printers require not using motion strings to move the current position to a margin and then fixing the margin there, but require the specification of where a margin should be regardless of the current position. Therefore terminfo offers six additional strings for defining margins with printers.

Setting Margins

smgl Set left margin at current column
smgr Set right margin at current column
smgb Set bottom margin at current line
smgt Set top margin at current line
smgbp Set bottom margin at line N
smglp Set left margin at line N
gmgrp Set right margin at column N
smgtp Set top margin at line N

man pages section 4: File Formats • Last Revised 9 Jul 1996
The last four strings are used with one or more arguments that give the position of the margin or margins to set. If both of smglp and smgrp are set, each is used with a single argument, \( N \), that gives the column number of the left and right margin, respectively. If both of smgtip and smgbp are set, each is used to set the top and bottom margin, respectively: smgtip is used with a single argument, \( N \), the line number of the top margin; however, smgbp is used with two arguments, \( N \) and \( M \), that give the line number of the bottom margin, the first counting from the top of the page and the second counting from the bottom. This accommodates the two styles of specifying the bottom margin in different manufacturers’ printers. When coding a \texttt{terminfo} entry for a printer that has a settable bottom margin, only the first or second parameter should be used, depending on the printer. When writing an application that uses smgbp to set the bottom margin, both arguments must be given.

If only one of smglp and smgrp is set, then it is used with two arguments, the column number of the left and right margins, in that order. Likewise, if only one of smgtip and smgbp is set, then it is used with two arguments that give the top and bottom margins, in that order, counting from the top of the page. Thus when coding a \texttt{terminfo} entry for a printer that requires setting both left and right or top and bottom margins simultaneously, only one of smglp and smgrp or smgtip and smgbp should be defined; the other should be left blank. When writing an application that uses these string capabilities, the pairs should be first checked to see if each in the pair is set or only one is set, and should then be used accordingly.

In counting lines or columns, line zero is the top line and column zero is the left-most column. A zero value for the second argument with smgbp means the bottom line of the page.

All margins can be cleared with mgc.

Five new sets of strings describe the capabilities printers have of enhancing printed text.

Enhanced Printing

\begin{itemize}
  \item sshm Enter shadow-printing mode
  \item rshm Exit shadow-printing mode
  \item sitm Enter italicizing mode
  \item ritm Exit italicizing mode
  \item swidm Enter wide character mode
  \item rwidm Exit wide character mode
  \item ssupm Enter superscript mode
  \item rsupm Exit superscript mode
  \item supcs List of characters available as superscripts
  \item ssubm Enter subscript mode
  \item rsubm Exit subscript mode
  \item subcs List of characters available as subscripts
\end{itemize}
If a printer requires the sshm control sequence before every character to be shadow-printed, the rshm string is left blank. Thus programs that find a control sequence in sshm but none in rshm should use the sshm control sequence before every character to be shadow-printed; otherwise, the sshm control sequence should be used once before the set of characters to be shadow-printed, followed by rshm. The same is also true of each of the sitm/ritm, swidm/rwidm, ssupm/rsupm, and sssubm/rsubm pairs.

Note that terminfo also has a capability for printing emboldened text (bold). While shadow printing and emboldened printing are similar in that they “darken” the text, many printers produce these two types of print in slightly different ways. Generally, emboldened printing is done by overstriking the same character one or more times. Shadow printing likewise usually involves overstriking, but with a slight movement up and/or to the side so that the character is “fatter.”

It is assumed that enhanced printing modes are independent modes, so that it would be possible, for instance, to shadow print italicized subscripts.

As mentioned earlier, the amount of motion automatically made after printing a wide character should be given in widcs.

If only a subset of the printable ASCII characters can be printed as superscripts or subscripts, they should be listed in supcs or subcs strings, respectively. If the ssupm or ssupm strings contain control sequences, but the corresponding supcs or subcs strings are empty, it is assumed that all printable ASCII characters are available as superscripts or subscripts.

Automatic motion made after printing a superscript or subscript is assumed to be the same as for regular characters. Thus, for example, printing any of the following three examples will result in equivalent motion:

\[
\text{Bi } B_i \text{ B}^i
\]

Note that the existing msgr boolean capability describes whether motion control sequences can be used while in “standout mode.” This capability is extended to cover the enhanced printing modes added here. msgr should be set for those printers that accept any motion control sequences without affecting shadow, italicized, widened, superscript, or subscript printing. Conversely, if msgr is not set, a program should end these modes before attempting any motion.

In addition to allowing you to define line graphics (described in Section 1-12), terminfo lets you define alternate character sets. The following capabilities cover printers and terminals with multiple selectable or definable character sets.

Alternate Character Sets

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scs</td>
<td>Select character set N</td>
</tr>
<tr>
<td>scsd</td>
<td>Start definition of character set N, M characters</td>
</tr>
<tr>
<td>defc</td>
<td>Define character A, B dots wide, descender D</td>
</tr>
</tbody>
</table>
The `scs`, `rcsd`, and `csnm` strings are used with a single argument, \( N \), a number from 0 to 63 that identifies the character set. The `scsd` string is also used with the argument \( N \) and another, \( M \), that gives the number of characters in the set. The `defc` string is used with three arguments: \( A \) gives the ASCII code representation for the character, \( B \) gives the width of the character in dots, and \( D \) is zero or one depending on whether the character is a “descender” or not. The `defc` string is also followed by a string of “image-data” bytes that describe how the character looks (see below).

Character set 0 is the default character set present after the printer has been initialized. Not every printer has 64 character sets, of course; using `scs` with an argument that doesn’t select an available character set should cause a null result from `tparm`.

If a character set has to be defined before it can be used, the `scsd` control sequence is to be used before defining the character set, and the `rcsd` is to be used after. They should also cause a null result from `tparm` when used with an argument \( N \) that doesn’t apply. If a character set still has to be selected after being defined, the `scs` control sequence should follow the `rcsd` control sequence. By examining the results of using each of the `scs`, `scsd`, and `rcsd` strings with a character set number in a call to `tparm`, a program can determine which of the three are needed.

Between use of the `scsd` and `rcsd` strings, the `defc` string should be used to define each character. To print any character on printers covered by `terminfo`, the ASCII code is sent to the printer. This is true for characters in an alternate set as well as “normal” characters. Thus the definition of a character includes the ASCII code that represents it. In addition, the width of the character in dots is given, along with an indication of whether the character should descend below the print line (such as the lower case letter “g” in most character sets). The width of the character in dots also indicates the number of image-data bytes that will follow the `defc` string. These image-data bytes indicate where in a dot-matrix pattern ink should be applied to “draw” the character; the number of these bytes and their form are defined below under “Dot-Mapped Graphics.”

It’s easiest for the creator of `terminfo` entries to refer to each character set by number; however, these numbers will be meaningless to the application developer. The `csnm` string alleviates this problem by providing names for each number.

When used with a character set number in a call to `tparm`, the `csnm` string will produce the equivalent name. These names should be used as a reference only. No naming convention is implied, although anyone who creates a `terminfo` entry for a printer should use names consistent with the names found in user documents for the printer. Application developers should allow a user to specify a character set by number (leaving it up to the user to examine the `csnm` string to determine the correct number), or by name, where the application examines the `csnm` string to determine the corresponding character set number.
These capabilities are likely to be used only with dot-matrix printers. If they are not available, the strings should not be defined. For printers that have manually changed print-wheels or font cartridges, the boolean daisy is set.

Dot-matrix printers typically have the capability of reproducing “raster-graphics” images. Three new numeric capabilities and three new string capabilities can help a program draw raster-graphics images independent of the type of dot-matrix printer or the number of pins or dots the printer can handle at one time.

### Dot-Matrix Graphics

- **npins**: Number of pins, \( N \), in print-head
- **spinv**: Spacing of pins vertically in pins per inch
- **spinh**: Spacing of dots horizontally in dots per inch
- **porder**: Matches software bits to print-head pins
- **sbim**: Start printing bit image graphics, \( B \) bits wide
- **rbim**: End printing bit image graphics

The `sbim` string is used with a single argument, \( B \), the width of the image in dots.

The model of dot-matrix or raster-graphics that `terminfo` presents is similar to the technique used for most dot-matrix printers: each pass of the printer’s print-head is assumed to produce a dot-matrix that is \( N \) dots high and \( B \) dots wide. This is typically a wide, squat, rectangle of dots. The height of this rectangle in dots will vary from one printer to the next; this is given in the `npins` numeric capability. The size of the rectangle in fractions of an inch will also vary; it can be deduced from the `spinv` and `spinh` numeric capabilities. With these three values an application can divide a complete raster-graphics image into several horizontal strips, perhaps interpolating to account for different dot spacing vertically and horizontally.

The `sbim` and `rbim` strings are used to start and end a dot-matrix image, respectively. The `sbim` string is used with a single argument that gives the width of the dot-matrix in dots. A sequence of “image-data bytes” are sent to the printer after the `sbim` string and before the `rbim` string. The number of bytes is an integral multiple of the width of the dot-matrix; the multiple and the form of each byte is determined by the `porder` string as described below.

The `porder` string is a comma separated list of pin numbers optionally followed by an numerical offset. The offset, if given, is separated from the list with a semicolon. The position of each pin number in the list corresponds to a bit in an 8-bit data byte. The pins are numbered consecutively from 1 to `npins`, with 1 being the top pin. Note that the term “pin” is used loosely here; “ink-jet” dot-matrix printers don’t have pins, but can be considered to have an equivalent method of applying a single dot of ink to paper. The bit positions in `porder` are in groups of 8, with the first position in each group the most significant bit and the last position the least significant bit. An application produces 8-bit bytes in the order of the groups in `porder`. 
An application computes the “image-data bytes” from the internal image, mapping vertical dot positions in each print-head pass into 8-bit bytes, using a 1 bit where ink should be applied and 0 where no ink should be applied. This can be reversed (0 bit for ink, 1 bit for no ink) by giving a negative pin number. If a position is skipped in porder, a 0 bit is used. If a position has a lower case ‘x’ instead of a pin number, a 1 bit is used in the skipped position. For consistency, a lower case ‘o’ can be used to represent a filled, skipped bit. There must be a multiple of 8 bit positions used or skipped in porder; if not, 0 bits are used to fill the last byte in the least significant bits. The offset, if given, is added to each data byte; the offset can be negative.

Some examples may help clarify the use of the porder string. The AT&T 470, AT&T 475 and C.Itoh 8510 printers provide eight pins for graphics. The pins are identified top to bottom by the 8 bits in a byte, from least significant to most. The porder strings for these printers would be 8, 7, 6, 5, 4, 3, 2, 1. The AT&T 478 and AT&T 479 printers also provide eight pins for graphics. However, the pins are identified in the reverse order. The porder strings for these printers would be 1, 2, 3, 4, 5, 6, 7, 8. The AT&T 5310, AT&T 5320, DEC LA100, and DEC LN03 printers provide six pins for graphics. The pins are identified top to bottom by the decimal values 1, 2, 4, 8, 16 and 32. These correspond to the low six bits in an 8-bit byte, although the decimal values are further offset by the value 63. The porder string for these printers would be 6, 5, 4, 3, 2, 1; 63, or alternately o, o, 6, 5, 4, 3, 2, 1; 63.

If the control sequences to change the character pitch or the line pitch are used, the pin or dot spacing may change:

**Dot-Matrix Graphics**

Changing the Character/Line Pitches

- **cpi** Change character pitch
- **cpix** If set, cpi changes spinh
- **lpi** Change line pitch
- **lpix** If set, lpi changes spinv

Programs that use cpi or lpi should recalculate the dot spacing:

**Dot-Matrix Graphics**

Effects of Changing the Character/Line Pitches

<table>
<thead>
<tr>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{spin}h$</td>
<td>$\text{spin}h$</td>
</tr>
</tbody>
</table>

Using cpi with cpix clear:

$\text{spin}h$ $\text{spin}h$

Using cpi with cpix set:

$\text{spin}h$ $\text{spin}h = \text{spin}h \cdot \text{dot orhi over} \{ \text{bold orhi} \}$

Using lpi with lpix clear:

$\text{spin}v$ $\text{spin}v$
Using lpi with lpx set:
\[ \text{bold spinv}' \cdot \text{bold spinv} = \text{bold spinv} \cdot \text{cdot} \cdot \text{bold orhi} \over \text{bold orhi}' \]

Using chr:
\[ \text{bold spinh}' \]

Using cvr:
\[ \text{bold spinv}' \]

orhi’ and orhi are the values of the horizontal resolution in steps per inch, before using cpi and after using cpi, respectively. Likewise, orvi’ and orvi are the values of the vertical resolution in steps per inch, before using lpi and after using lpi, respectively. Thus, the changes in the dots per inch for dot-matrix graphics follow the changes in steps per inch for printer resolution.

Many dot-matrix printers can alter the dot spacing of printed text to produce near “letter quality” printing or “draft quality” printing. Usually it is important to be able to choose one or the other because the rate of printing generally falls off as the quality improves. There are three new strings used to describe these capabilities.

Print Quality

- snlq: Set near-letter quality print
- snrmq: Set normal quality print
- sdrfq: Set draft quality print

The capabilities are listed in decreasing levels of quality. If a printer doesn’t have all three levels, one or two of the strings should be left blank as appropriate.

Because there is no standard protocol that can be used to keep a program synchronized with a printer, and because modern printers can buffer data before printing it, a program generally cannot determine at any time what has been printed. Two new numeric capabilities can help a program estimate what has been printed.

Print Rate/Buffer Size

- cps: Nominal print rate in characters per second
- bufsz: Buffer capacity in characters

cps is the nominal or average rate at which the printer prints characters; if this value is not given, the rate should be estimated at one-tenth the prevailing baud rate. bufsz is the maximum number of subsequent characters buffered before the guaranteed printing of an earlier character, assuming proper flow control has been used. If this value is not given it is assumed that the printer does not buffer characters, but prints them as they are received.
As an example, if a printer has a 1000-character buffer, then sending the letter “a” followed by 1000 additional characters is guaranteed to cause the letter “a” to print. If the same printer prints at the rate of 100 characters per second, then it should take 10 seconds to print all the characters in the buffer, less if the buffer is not full. By keeping track of the characters sent to a printer, and knowing the print rate and buffer size, a program can synchronize itself with the printer.

Note that most printer manufacturers advertise the maximum print rate, not the nominal print rate. A good way to get a value to put in for cps is to generate a few pages of text, count the number of printable characters, and then see how long it takes to print the text.

Applications that use these values should recognize the variability in the print rate. Straight text, in short lines, with no embedded control sequences will probably print at close to the advertised print rate and probably faster than the rate in cps. Graphics data with a lot of control sequences, or very long lines of text, will print at well below the advertised rate and below the rate in cps. If the application is using cps to decide how long it should take a printer to print a block of text, the application should pad the estimate. If the application is using cps to decide how much text has already been printed, it should shrink the estimate. The application will thus err in favor of the user, who wants, above all, to see all the output in its correct place.

FILES
/usr/share/lib/terminfo/?/* compiled terminal description database
/usr/share/lib/.COREterm/?/* subset of compiled terminal description database
/usr/share/lib/tabset/* tab settings for some terminals, in a format appropriate to be output to the terminal (escape sequences that set margins and tabs)

SEE ALSO
ls(1), pg(l), stty(l), tput(l), tty(l), vi(l), infocmp(lM), tic(lM), printf(3C), curses(3CURSES), curses(3XCURSES)

NOTES
The most effective way to prepare a terminal description is by imitating the description of a similar terminal in terminfo and to build up a description gradually, using partial descriptions with a screen oriented editor, such as vi, to check that they are correct. To easily test a new terminal description the environment variable TERMINFO can be set to the pathname of a directory containing the compiled description, and programs will look there rather than in /usr/share/lib/terminfo.
### NAME
TIMEZONE – set default system time zone and locale

### SYNOPSIS
/etc/TIMEZONE /etc/default/init

### DESCRIPTION
This file sets the time zone environment variable TZ, and the locale-related environment variables LANG, LC_COLLATE, LC_CTYPE, LC_MESSAGES, LC_MONETARY, LC_NUMERIC, and LC_TIME.

/etc/TIMEZONE is a symbolic link to /etc/default/init.

The number of environments that can be set from /etc/default/init is limited to 20.

### SEE ALSO
init(1M), ctime(3C), environ(5)
### NAME
timezone – default timezone data base

### SYNOPSIS
/etc/timezone

### DESCRIPTION
The timezone file contains information regarding the default timezone for each host in a domain. Alternatively, a single default line for the entire domain may be specified. Each entry has the format:

```
Timezone-name official-host-or-domain-name
```

Items are separated by any number of blanks and/or TAB characters. A ‘#’ indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file. The timezone is a pathname relative to the directory /usr/share/lib/zoneinfo.

This file is not actually referenced by any system software; it is merely used as a source file to construct the NIS timezone.bynmame map. This map is read by the program /usr/etc/install/sysIDtool to initialize the timezone of the client system at installation time.

The timezone file does not set the timezone environment variable TZ. See TIMEZONE(4) for information to set the TZ environment variable.

### EXAMPLES
**EXAMPLE 1** A sample display of timezone command.

Here is a typical line from the /etc/timezone file:

```
US/Eastern East.Sun.COM #Sun East Coast
```

### FILES
/etc/timezone

### SEE ALSO
TIMEZONE(4)
**NAME**
tnf_kernel_probes – TNF kernel probes

**DESCRIPTION**
The set of probes (trace instrumentation points) available in the standard kernel. The probes log trace data to a kernel trace buffer in Trace Normal Form (TNF). Kernel probes are controlled by `prex(1)`. A snapshot of the kernel trace buffer can be made using `tnfxtract(1)` and examined using `tnfdump(1)`.

Each probe has a *name* and is associated with a set of symbolic *keys*, or *categories*. These are used to select and control probes from `prex(1)`. A probe that is enabled for tracing generates a TNF record, called an *event record*. An event record contains two common members and may contain other probe-specific data members.

**Common Members**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tnf_probe_event</code></td>
<td><code>tag</code></td>
</tr>
<tr>
<td><code>tnf_time_delta</code></td>
<td><code>time_delta</code></td>
</tr>
</tbody>
</table>

**Threads**

**thread_create**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tnf_kthread_id</code></td>
<td><code>tid</code></td>
</tr>
<tr>
<td><code>tnf_pid</code></td>
<td><code>pid</code></td>
</tr>
<tr>
<td><code>tnf_symbol</code></td>
<td><code>start_pc</code></td>
</tr>
</tbody>
</table>

**thread_state**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tnf_kthread_id</code></td>
<td><code>tid</code></td>
</tr>
<tr>
<td><code>tnf_microstate</code></td>
<td><code>state</code></td>
</tr>
<tr>
<td></td>
<td>■ Running in user mode.</td>
</tr>
<tr>
<td></td>
<td>■ Running in system mode.</td>
</tr>
<tr>
<td></td>
<td>■ Asleep waiting for a user-mode lock.</td>
</tr>
</tbody>
</table>
Asleep on a kernel object.
- Runnable (waiting for a cpu).
- Stopped.

The values of this member are defined in `<sys/msacct.h>`. Note that to reduce trace output, transitions between the system and user microstates that are induced by system calls are not traced. This information is implicit in the system call entry and exit events.

**thread_exit**

Thread termination event for writing thread. This probe has no data members other than the common members.

**Scheduling**

**thread_queue**

```
tnf_kthread_id tid
tnf_cpuid cpuid
tnf_long priority
tnf_ulong queue_length
```

Thread scheduling events. These are triggered when a runnable thread is placed on a dispatch queue.

- `cpuid` Specifies the cpu to which the queue is attached.
- `priority` The (global) dispatch priority of the thread.
- `queue_length` The current length of the cpu’s dispatch queue.

**Blocking**

**thread_block**

```
tnf_opaque reason
tnf_symbols stack
```

Thread blockage event. This probe captures a partial stack backtrace when the current thread blocks.

- `reason` The address of the object on which the thread is blocking.
- `symbols` References a TNF array of kernel addresses representing the PCs on the stack at the time the thread blocks.

**System Calls**

**syscall_start**

```
tnf_sysnum sysnum
```

System call entry event.

- `sysnum` The system call number. The writing thread implicitly enters the system microstate with this event.
syscall_end

```c
tnf_long  rval1
tnf_long  rval2
tnf_long  errno
```

System call exit event.

`rval1` and `rval2` The two return values of the system call

`errno` The error return.

The writing thread implicitly enters the `user` microstate with this event.

**Page Faults**

**address_fault**

```c
tnf_opaque  address
tnf_fault_type  fault_type
tnf_seg_access  access
```

Address-space fault event.

`address` Gives the faulting virtual address.

`fault_type` Gives the fault type: invalid page, protection fault, software requested locking or unlocking.

`access` Gives the desired access protection: read, write, execute or create.

The values for these two members are defined in `<vm/seg_enum.h>`.

**major_fault**

```c
tnf_opaque  vnode
tnf_offset  offset
```

Major page fault event. The faulting page is mapped to the file given by the `vnode` member, at the given `offset` into the file. (The faulting virtual address is in the most recent `address_fault` event for the writing thread.)

**anon_private**

```c
tnf_opaque  address
```

Copy-on-write page fault event.

`address` The virtual address at which the new page is mapped.

**anon_zero**

```c
tnf_opaque  address
```

Zero-fill page fault event.

`address` The virtual address at which the new page is mapped.
Page unmapping event. This probe marks the unmapping of a file system page from the system.

\[ vnodo \text{ and } offset \]
Identifies the file and offset of the page being unmapped.

Pagein start event. This event signals the initiation of pagein I/O.

\[ vnodo \text{ and } offset \]
Identifies the file and offset to be paged in.

\[ size \]
Specifies the number of bytes to be paged in.

Pageout completion event. This event signals the completion of pageout I/O.

\[ vnodo \]
Identifies the file of the pageout request.

\[ pages\_pageout \]
The number of pages written out.

\[ pages\_freed \]
The number of pages freed after being written out.

\[ pages\_reclaimed \]
The number of pages reclaimed after being written out.

Page daemon scan start event. This event signals the beginning of one iteration of the page daemon.

\[ pages\_free \]
The number of free pages in the system.

\[ pages\_needed \]
The number of pages desired free.
### Pages

#### Page daemon scan end event

This event signals the end of one iteration of the page daemon.

- **pages_free**: The number of free pages in the system.
- **pages_scanned**: The number of pages examined by the page daemon. (Potentially more pages will be freed when any queued pageout requests complete.)

### Swapper

#### swapout_process

This event marks the swapping out of a process address space.

- **pid**: The process identifier.
- **page_count**: Reports the number of pages either freed or queued for pageout.

#### swapout_lwp

This event marks the swapping out of an LWP and its stack.

- **pid**: The LWP’s process identifier.
- **lwpid**: The LWP identifier.
- **tid**: The LWP’s kernel thread identifier.
- **page_count**: The number of pages swapped out.

#### swapin_lwp

This event marks the swapping in of an LWP and its stack.

- **pid**: The LWP’s process identifier.
- **lwpid**: The LWP identifier.
tid                      The LWP’s kernel thread identifier.

page_count               The number of pages swapped in.

strategy

| tnf_device   | device       |
| tnf_diskaddr  | block        |
| tnf_size     | size         |
| tnfOpaque    | buf          |
| tnf_bioflags | flags        |

Block I/O strategy event. This event marks a call to the strategy(9E) function of a block device driver.

| device       | Contains the major and minor numbers of the device. |
| block        | The logical block number to be accessed on the device. |
| size         | The size of the I/O request. |
| buf          | The kernel address of the buf(9S) structure associated with the transfer. |
| flags        | The buf(9S) flags associated with the transfer. |

biodone

| tnf_device   | device       |
| tnf_diskaddr  | block        |
| tnfOpaque    | buf          |

Buffered I/O completion event. This event marks calls to the biodone(9F) function.

| device       | Contains the major and minor numbers of the device. |
| block        | The logical block number accessed on the device. |
| buf          | The kernel address of the buf(9S) structure associated with the transfer. |

physio_start

| tnf_device   | device       |
| tnf_offset   | offset       |
| tnf_size     | size         |
| tnf_bioflags | rw           |

Raw I/O start event. This event marks entry into the physio(9F) function which performs unbuffered I/O.

| device       | Contains the major and minor numbers of the device of the transfer. |
| offset       | The logical offset on the device for the transfer. |
### tnf_kernel_probes(4)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>size</code></td>
<td>The number of bytes to be transferred.</td>
</tr>
<tr>
<td><code>rw</code></td>
<td>The direction of the transfer: read or write (see <code>buf(9S)</code>).</td>
</tr>
</tbody>
</table>

#### physio_end

`tnf_device  device`

Raw I/O end event. This event marks exit from the `physio(9F)` function.

`device`  The major and minor numbers of the device of the transfer.

### USAGE

Use the `prex` utility to control kernel probes. The standard `prex` commands to list and manipulate probes are available to you, along with commands to set up and manage kernel tracing.

Kernel probes write trace records into a kernel trace buffer. You must copy the buffer into a TNF file for post-processing; use the `tnfxtract` utility for this.

You use the `tnfdump` utility to examine a kernel trace file. This is exactly the same as examining a user-level trace file.

The steps you typically follow to take a kernel trace are:

1. Become superuser (`su`).
2. Allocate a kernel trace buffer of the desired size (`prex`).
3. Select the probes you want to trace and enable (`prex`).
4. Turn kernel tracing on (`prex`).
5. Run your application.
6. Turn kernel tracing off (`prex`).
7. Extract the kernel trace buffer (`tnfxtract`).
8. Disable all probes (`prex`).
9. Deallocate the kernel trace buffer (`prex`).
10. Examine the trace file (`tnfdump`).

A convenient way to follow these steps is to use two shell windows; run an interactive `prex` session in one, and run your application and `tnfxtract` in the other.

### SEE ALSO

`prex(1), tnfdump(1), tnfxtract(1), libtnfctl(3TNF), TNF_PROBE(3TNF), tracing(3TNF), strategy(9E), biodone(9F), physio(9F), buf(9S)`
The process scheduler (or dispatcher) is the portion of the kernel that controls allocation of the CPU to processes. The scheduler supports the notion of scheduling classes where each class defines a scheduling policy, used to schedule processes within that class. Associated with each scheduling class is a set of priority queues on which ready to run processes are linked. These priority queues are mapped by the system configuration into a set of global scheduling priorities which are available to processes within the class. (The dispatcher always selects for execution the process with the highest global scheduling priority in the system.) The priority queues associated with a given class are viewed by that class as a contiguous set of priority levels numbered from 0 (lowest priority) to $n$ (highest priority—a configuration-dependent value). The set of global scheduling priorities that the queues for a given class are mapped into might not start at zero and might not be contiguous (depending on the configuration).

Processes in the time-sharing class which are running in user mode (or in kernel mode before going to sleep) are scheduled according to the parameters in a time-sharing dispatcher parameter table ($ts_dptbl$). Processes in the inter-active scheduling class are also scheduled according to the parameters in the time-sharing dispatcher parameter table. (Time-sharing processes and inter-active processes running in kernel mode after sleeping are run within a special range of priorities reserved for such processes and are not affected by the parameters in the $ts_dptbl$ until they return to user mode.) The $ts_dptbl$ consists of an array ($config_ts_dptbl[]$) of parameter structures ($struct tsdpent_t$), one for each of the $n$ priority levels used by time-sharing processes and inter-active processes in user mode. The structures are accessed via a pointer ($ts_dptbl$), to the array. The properties of a given priority level $i$ are specified by the $i$th parameter structure in this array ($ts_dptbl[i]$).

A parameter structure consists of the following members. These are also described in the /usr/include/sys/ts.h header.

- $ts_globpri$: The global scheduling priority associated with this priority level. The mapping between time-sharing priority levels and global scheduling priorities is determined at boot time by the system configuration. $ts_globpri$ is the only member of the $ts_dptbl$ which cannot be changed with dispadmin(1M).

- $ts_quantum$: The length of the time quantum allocated to processes at this level in ticks (Hz).

- $ts_tqexp$: Priority level of the new queue on which to place a process running at the current level if it exceeds its time quantum. Normally this field links to a lower priority time-sharing level that has a larger quantum.

- $ts_slpret$: Priority level of the new queue on which to place a process, that was previously in user mode at this level,
when it returns to user mode after sleeping. Normally
this field links to a higher priority level that has a
smaller quantum.

**ts_maxwait**

A per process counter, ts_dispwait is initialized to
zero each time a time-sharing or inter-active process is
placed back on the dispatcher queue after its time
quantum has expired or when it is awakened
(ts_dispwait is not reset to zero when a process is
preempted by a higher priority process). This counter is
incremented once per second for each process on the
dispatcher queue. If a process’s ts_dispwait value
exceeds the ts_maxwait value for its level, the
process’s priority is changed to that indicated by
ts_lwait. The purpose of this field is to prevent
starvation.

**ts_lwait**

Move a process to this new priority level if
ts_dispwait is greater than ts_maxwait.

An administrator can affect the behavior of the time-sharing portion of the scheduler
by reconfiguring the ts_dptbl. Since processes in the time-sharing and inter-active
scheduling classes share the same dispatch parameter table (ts_dptbl), changes to
this table will affect both scheduling classes. There are two methods available for
doing this: reconfigure with a loadable module at boot-time or by using
dispadmin(1M) at run-time.

**ts_dptbl Loadable Module**

The ts_dptbl can be reconfigured with a loadable module which contains a new
time sharing dispatch table. The module containing the dispatch table is separate from
the TS loadable module which contains the rest of the time-sharing and inter-active
software. This is the only method that can be used to change the number of
time-sharing priority levels or the set of global scheduling priorities used by the
time-sharing and inter-active classes. The relevant procedure and source code is
described in the REPLACING THE TS_DPTBL LOADABLE MODULE section.

**dispadmin Configuration File**

With the exception of ts_globpri all of the members of the ts_dptbl can be
examined and modified on a running system using the dispadmin(1M) command.
Invoking dispadmin for the time-sharing or inter-active class allows the
administrator to retrieve the current ts_dptbl configuration from the kernel’s in-core
table, or overwrite the in-core table with values from a configuration file. The
configuration file used for input to dispadmin must conform to the specific format
described below.

Blank lines are ignored and any part of a line to the right of a # symbol is treated as a
comment. The first non-blank, non-comment line must indicate the resolution to be
used for interpreting the ts_quantum time quantum values. The resolution is
specified as

RES=res
where \( res \) is a positive integer between 1 and 1,000,000,000 inclusive and the resolution used is the reciprocal of \( res \) in seconds (for example, \( RES=1000 \) specifies millisecond resolution). Although very fine (nanosecond) resolution may be specified, the time quantum lengths are rounded up to the next integral multiple of the system clock’s resolution.

The remaining lines in the file are used to specify the parameter values for each of the time-sharing priority levels. The first line specifies the parameters for time-sharing level 0, the second line specifies the parameters for time-sharing level 1, etc. There must be exactly one line for each configured time-sharing priority level.

**EXAMPLE 1** A Sample From a Configuration File

The following excerpt from a `dispadmin` configuration file illustrates the format. Note that for each line specifying a set of parameters there is a comment indicating the corresponding priority level. These level numbers indicate priority within the time-sharing and interactive classes, and the mapping between these time-sharing priorities and the corresponding global scheduling priorities is determined by the configuration specified in the `ts` master file. The level numbers are strictly for the convenience of the administrator reading the file and, as with any comment, they are ignored by `dispadmin`. `dispadmin` assumes that the lines in the file are ordered by consecutive, increasing priority level (from 0 to the maximum configured time-sharing priority). The level numbers in the comments should normally agree with this ordering; if for some reason they don’t, however, `dispadmin` is unaffected.

```plaintext
# Time-Sharing Dispatcher Configuration File RES=1000

# ts_quantum ts_tqexp ts_slpret ts_maxwait ts_lwait PRIORITY
# LEVEL
500 0 10 5 10 # 0
500 0 11 5 11 # 1
500 1 12 5 12 # 2
500 1 13 5 13 # 3
500 2 14 5 14 # 4
500 2 15 5 15 # 5
450 3 16 5 16 # 6
450 3 17 5 17 # 7
450 3 18 5 18 # 8
50 48 59 5 59 # 58
50 49 59 5 59 # 59
```

**EXAMPLE 2** Replacing The `ts_dptbl` Loadable Module

In order to change the size of the time sharing dispatch table, the loadable module which contains the dispatch table information will have to be built. It is recommended that you save the existing module before using the following procedure.
**EXAMPLE 2 Replacing The ts_dptbl Loadable Module**  
*(Continued)*

1. Place the dispatch table code shown below in a file called `ts_dptbl.c`. An example of this file follows.

2. Compile the code using the given compilation and link lines supplied.

   ```
   cc -c -O -D_KERNEL  
   ts_dptbl.c  
   ld -r -o TS_DPTBL ts_dptbl.o
   ```

3. Copy the current dispatch table in `/kernel/sched` to `/kernel/sched/TS_DPTBL.bak`.

4. Replace the current `TS_DPTBL` in `/kernel/sched`.

5. You will have to make changes in the `/etc/system` file to reflect the changes to the sizes of the tables. See `system(4)`. The two variables affected are `ts_maxupri` and `ts_maxkmdpri`. The syntax for setting these is as follows:

   ```
   set TS:ts_maxupri=(value for max time-sharing user priority)  
   set TS:ts_maxkmdpri=(number of kernel mode priorities - 1)
   ```

6. Reboot the system to use the new dispatch table.

Great care should be used in replacing the dispatch table using this method. If you do not get it right, panics may result, thus making the system unusable.

The following is an example of a `ts_dptbl.c` file used for building the new `ts_dptbl`.

```c
/* BEGIN ts_dptbl.c */
#include <sys/proc.h>
#include <sys/priocntl.h>
#include <sys/class.h>
#include <sys/disp.h>
#include <sys/ts.h>
#include <sys/rtpriocntl.h>
/*
* This is the loadable module wrapper.
*/
#include <sys/modctl.h>
extern struct mod_ops mod_miscops;
/*
* Module linkage information for the kernel.
*/
static struct modlmisc modlmisc = {
   &mod_miscops, "Time sharing dispatch table"},
static struct modlinkage modlinkage = {
   MODREV_1, &modlmisc, 0
};
_init()
{
   return (mod_install(&modlinkage));
}_info(modinfop)
   struct modinfo *modinfop;
```
EXAMPLE 2 Replacing The ts_dptbl Loadable Module  (Continued)

```c
{  return (mod_info(&modlinkage, modinfop));
}

/*
 * array of global priorities used by ts procs sleeping or
 * running in kernel mode after sleep. Must have at least
 * 40 values.
 */

pri_t config_ts_kmdpris[] = { 60, 61, 62, 63, 64, 65, 66, 67, 68, 69,
    70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
    80, 81, 82, 83, 84, 85, 86, 87, 88, 89,
    90, 91, 92, 93, 94, 95, 96, 97, 98, 99,
};
tsdpent_t config_ts_dptbl[] = {

/* glbpri qntm tqexp slprt mxwt lwt */
    0,   100, 0,   10, 5, 10,
    1,   100, 0,   11, 5, 11,
    2,   100, 1,   12, 5, 12,
    3,   100, 1,   13, 5, 13,
    4,   100, 2,   14, 5, 14,
    5,   100, 2,   15, 5, 15,
    6,   100, 3,   16, 5, 16,
    7,   100, 3,   17, 5, 17,
    8,   100, 4,   18, 5, 18,
    9,   100, 4,   19, 5, 19,
   10,   80, 5,   20, 5, 20,
   11,   80, 5,   21, 5, 21,
   12,   80, 6,   22, 5, 22,
   13,   80, 6,   23, 5, 23,
   14,   80, 7,   24, 5, 24,
   15,   80, 7,   25, 5, 25,
   16,   80, 8,   26, 5, 26,
   17,   80, 8,   27, 5, 27,
   18,   80, 9,   28, 5, 28,
   19,   80, 9,   29, 5, 29,
   20,   60, 10,  30, 5, 30,
   21,   60, 11,  31, 5, 31,
   22,   60, 12,  32, 5, 33,
   24,   60, 14,  34, 5, 34,
   25,   60, 15,  35, 5, 35,
   26,   60, 16,  36, 5, 36,
   27,   60, 17,  37, 5, 37,
   28,   60, 18,  38, 5, 38,
   29,   60, 19,  39, 5, 39,
   30,   40, 20,  40, 5, 40,
   31,   40, 21,  41, 5, 41,
   32,   40, 22,  42, 5, 42,
   33,   40, 23,  43, 5, 43,
   34,   40, 24,  44, 5, 44,
   35,   40, 25,  45, 5, 45,
```
EXAMPLE 2 Replacing The ts_dptbl Loadable Module (Continued)

```c
};
short config_ts_maxumdpri = sizeof (config_ts_dptbl)/16 - 1;
/*
 * Return the address of config_ts_dptbl
 */
tsdpent_t *
ts_getdptbl()
{
    return (config_ts_dptbl);
}

/*
 * Return the address of config_ts_kmdpris
 */
int *
ts_getkmdpris()
{
    return (config_ts_kmdpris);
}

/*
 * Return the address of ts_maxumdpri
 */
short
ts_getmaxumdpri()
{
    return (config_ts_maxumdpri);
}
```
EXAMPLE 2 Replacing The ts_dptbl Loadable Module (Continued)

/* END ts_dptbl.c */

FILES

<sys/ts.h>

SEE ALSO

priocntl(1), dispadmin(1M), priocntl(2), system(4)

System Administration Guide, Volume 1

System Interface Guide

NOTES

dispadmin does some limited sanity checking on the values supplied in the configuration file. The sanity checking is intended to ensure that the new ts_dptbl values do not cause the system to panic. The sanity checking does not attempt to analyze the effect that the new values will have on the performance of the system. Unusual ts_dptbl configurations may have a dramatic negative impact on the performance of the system.

No sanity checking is done on the ts_dptbl values specified in the TS_DPTBL loadable module. Specifying an inconsistent or nonsensical ts_dptbl configuration through the TS_DPTBL loadable module could cause serious performance problems and/or cause the system to panic.
ttydefs(4)

NAME

ttydefs – file contains terminal line settings information for ttymon

DESCRIPTION

/etc/ttydefs is an administrative file that contains records divided into fields by
colons (":"). This information used by ttymon to set up the speed and terminal
settings for a TTY port.

The ttydefs file contains the following fields:

- **ttylabel**
  The string ttymon tries to match against the TTY port’s ttylabel
  field in the port monitor administrative file. It often describes the
  speed at which the terminal is supposed to run, for example, 1200.

- **initial-flags**
  Contains the initial termio(7I) settings to which the terminal is to
  be set. For example, the system administrator will be able to
  specify what the default erase and kill characters will be. initial-flags
  must be specified in the syntax recognized by the stty command.

- **final-flags**
  final-flags must be specified in the same format as initial-flags.
  ttymon sets these final settings after a connection request has been
  made and immediately prior to invoking a port’s service.

- **autobaud**
  If the autobaud field contains the character ‘A,’ autobaud will be
  enabled. Otherwise, autobaud will be disabled. ttymon
determines what line speed to set the TTY port to by analyzing the
  carriage returns entered. If autobaud has been disabled, the hunt
  sequence is used for baud rate determination.

- **nextlabel**
  If the user indicates that the current terminal setting is not
  appropriate by sending a BREAK, ttymon searches for a ttydefs
  entry whose ttylabel field matches the nextlabel field. If a match is
  found, ttymon uses that field as its ttylabel field. A series of speeds
  is often linked together in this way into a closed set called a hunt
  sequence. For example, 4800 may be linked to 1200, which in
  turn is linked to 2400, which is finally linked to 4800.

SEE ALSO

sttydefs(1M), ttymon(1M), termio(7I)

System Administration Guide, Volume 1
ttysrch(4)

NAME

ttysrch – directory search list for ttyname

DESCRIPTION

ttysrch is an optional file that is used by the ttynamer library routine. This file contains the names of directories in /dev that contain terminal and terminal-related device files. The purpose of this file is to improve the performance of ttynamer by indicating which subdirectories in /dev contain terminal-related device files and should be searched first. These subdirectory names must appear on separate lines and must begin with /dev. Those path names that do not begin with /dev will be ignored and a warning will be sent to the console. Blank lines (lines containing only white space) and lines beginning with the comment character “#” will be ignored. For each file listed (except for the special entry /dev), ttynamer will recursively search through subdirectories looking for a match. If /dev appears in the ttysrch file, the /dev directory itself will be searched but there will not be a recursive search through its subdirectories.

When ttynamer searches through the device files, it tries to find a file whose major/minor device number, file system identifier, and inode number match that of the file descriptor it was given as an argument. If a match is not found, it will settle for a match of just major/minor device and file system identifier, if one can be found. However, if the file descriptor is associated with a clonable device, this algorithm does not work efficiently because the inode number of the device file associated with a clonable device will never match the inode number of the file descriptor that was returned by the open of that clonable device. To help with these situations, entries can be put into the /etc/ttysrch file to improve performance when cloned devices are used as terminals on a system (for example, for remote login). However, this is only useful if the minor devices related to a clonable device are put into a subdirectory. (It is important to note that device files need not exist for cloned devices and if that is the case, ttynamer will eventually fail.) An optional second field is used in the /etc/ttysrch file to indicate the matching criteria. This field is separated by white space (any combination of blanks or tabs). The letter M means major/minor device number, F means file system identifier, and I means inode number. If this field is not specified for an entry, the default is MFI which means try to match on all three. For cloned devices the field should be MF, which indicates that it is not necessary to match on the inode number.

Without the /etc/ttysrch file, ttynamer will search the /dev directory by first looking in the directories /dev/term, /dev/pts, and /dev/xt. If a system has terminal devices installed in directories other than these, it may help performance if the ttysrch file is created and contains that list of directories.

EXAMPLES

EXAMPLE 1 A sample display of /etc/ttysrch command.

A sample /etc/ttysrch file follows:

/dev/term   MFI
/dev/pts    MFI
/dev/xt     MFI
/dev/slan   MF
EXAMPLE 1 A sample display of /etc/ttysrch command.  (Continued)

This file tells ttyname that it should first search through those directories listed and that when searching through the /dev/scan directory, if a file is encountered whose major/minor devices and file system identifier match that of the file descriptor argument to ttyname, this device name should be considered a match.

FILES
/etc/ttysrch

SEE ALSO
ttyname(3C)
NAME| ufsdump, dumpdates – incremental dump format
SYNOPSIS| 

```c
#include <sys/types.h>
#include <sys/inode.h>
#include <protocols/dumprestore.h>
/etc/dumpdates
```

DESCRIPTION| Tapes used by ufsdump(1M) and ufsrestore(1M) contain:
- a header record
- two groups of bitmap records
- a group of records describing directories
- a group of records describing files

The format of the header record and of the first record of each description as given in the include file `<protocols/dumprestore.h>` is:

```c
#define TP_BSIZE 1024
#define NTREC 10
#define HIGHDENSITYTREC 32
#define CARTRIDGETREC 63
#define TP_NINDIR (TP_BSIZE/2)
#define TP_NINOS (TP_NINDIR / sizeop (long))
#define LBLSIZE 16
#define NAMELEN 64

#define NFS_MAGIC (int) 60012
#define CHECKSUM (int) 84446
```

```c
union u_data {
    char s_addrs[TP_NINDIR];
    long s_inos[TP_NINOS];
};
union u_spcl {
    char dummy[TP_BSIZE];
    struct s_spcl {
        long c_type;
        time_t c_date;
        time_t c_ddate;
        long c_volume;
        daddr_t c_tapea;
        ino_t c_inumber;
        long c_magic;
        long c_checksum;
        struct dinode c_dinode;
    }
}
```

ufsdump(4)

long c_count;
union u_data c_data;
char c_label[LBLSIZE];
long c_level;
char c_filesys[NAMELEN];
char c_dev[NAMELEN];
char c_host[NAMELEN];
long c_flags;
long c_firstrec;
long c_spare[32];
} s_spcl;
} u_spcl;

long c_type;
time_t c_date;
time_t c_ddate;
long c_volume;
daddr_t c_tapea;
ino_t c_inumber;
long c_magic;
long c_checksum;
struct dinode c_dinode;
long c_count;
union u_data c_data;
char c_label[LBLSIZE];
long c_level;
char c_filesys[NAMELEN];
char c_dev[NAMELEN];
char c_host[NAMELEN];
long c_flags;
long c_firstrec;
long c_spare[32];

} s_spcl;
} u_spcl;
#define spcl u_spcl.s_spcl
#define c_addr c_data.s_addr
#define c_inos cdata.s_inos

ufsdump(4)
546
man pages section 4: File Formats • Last Revised 7 Jan 1994
The constants are described as follows:

**TP_BSIZE**
Size of file blocks on the dump tapes. Note that **TP_BSIZE** must be a multiple of **DEV_BSIZE**.

**NTREC**
Default number of **TP_BSIZE** byte records in a physical tape block, changeable by the **b** option to **ufsdump(1M)**.

**HIGHDENSITYNTREC**
Default number of **TP_BSIZE** byte records in a physical tape block on 6250 BPI or higher density tapes.

**CARTRIDGETREC**
Default number of **TP_BSIZE** records in a physical tape block on cartridge tapes.

**TP_NINDIR**
Number of indirect pointers in a **TS_INODE** or **TS_ADDR** record. It must be a power of 2.

**TP_NINOS**
The maximum number of volumes on a tape. Used for tape labeling in **hsmdump** and **hsmrestore** (available with Online:Backup 2.0 optional software package **SUNWhsm**).

**LBLSIZE**
The maximum size of a volume label. Used for tape labeling in **hsmdump** and **hsmrestore** (available with Online:Backup 2.0 optional software package **SUNWhsm**).
NAMELEN
  The maximum size of a host’s name.

NFS_MAGIC
  All header records have this number in c_magic.

CHECKSUM
  Header records checksum to this value.

The TS_entries are used in the c_type field to indicate what sort of header this is. The types and their meanings are as follows:

<table>
<thead>
<tr>
<th>Entry</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS_TAPE</td>
<td>Tape volume label</td>
</tr>
<tr>
<td>TS_INODE</td>
<td>A file or directory follows. The c_dinode field is a copy of the disk inode and contains bits telling what sort of file this is.</td>
</tr>
<tr>
<td>TS_ADDR</td>
<td>A subrecord of a file description. See s_addrs below.</td>
</tr>
<tr>
<td>TS_BITS</td>
<td>A bit map follows. This bit map has a one bit for each inode that was dumped.</td>
</tr>
<tr>
<td>TS_CLRI</td>
<td>A bit map follows. This bit map contains a zero bit for all inodes that were empty on the file system when dumped.</td>
</tr>
<tr>
<td>TS_END</td>
<td>End of tape record.</td>
</tr>
<tr>
<td>TS_EOM</td>
<td>floppy EOM — restore compat with old dump</td>
</tr>
</tbody>
</table>

The flags are described as follows:

<table>
<thead>
<tr>
<th>Flag</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DR_NEWHEADER</td>
<td>New format tape header.</td>
</tr>
<tr>
<td>DR_INFODEINFO</td>
<td>Header contains starting inode info.</td>
</tr>
<tr>
<td>DR_REDUMP</td>
<td>Dump contains recopies of active files.</td>
</tr>
<tr>
<td>DR_TRUEINC</td>
<td>Dump is a &quot;true incremental&quot;.</td>
</tr>
<tr>
<td>DUMPOUTFMT</td>
<td>Name, incon, and ctime (date) for printf.</td>
</tr>
<tr>
<td>DUMPPINFMT</td>
<td>Inverse for scanf.</td>
</tr>
</tbody>
</table>

The fields of the header structure are as follows:

<table>
<thead>
<tr>
<th>Field</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_addrs</td>
<td>An array of bytes describing the blocks of the dumped file. A byte is zero if the block associated with that byte was not present on the file system; otherwise, the byte is non-zero. If the block was not present on the file system, no block was dumped; the block will be stored as a hole in the file. If there is not sufficient space in this record to describe all the blocks in a file, TS_ADDR records will be scattered through the file, each one picking up where the last left off</td>
</tr>
<tr>
<td>s_inos</td>
<td>The starting inodes on tape.</td>
</tr>
<tr>
<td>c_type</td>
<td>The type of the record.</td>
</tr>
<tr>
<td>c_date</td>
<td>The date of the previous dump.</td>
</tr>
</tbody>
</table>
The dump history is kept in the file /etc/dumpdates. It is an ASCII file with three fields separated by white space:

- The name of the device on which the dumped file system resides.
- The level number of the dump tape; see ufsdump(1M).
- The date of the incremental dump in the format generated by ctime(3C).

DUMPOUTFMT is the format to use when using printf(3C) to write an entry to /etc/dumpdates; DUMPINFMT is the format to use when using scanf(3C) to read an entry from /etc/dumpdates.

**ATTRIBUTES**

See attributes(5) for a description of the following attributes:
ufsdump(4)

<table>
<thead>
<tr>
<th>ATTRIBUTE TYPE</th>
<th>ATTRIBUTE VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stability Level</td>
<td>Unstable</td>
</tr>
</tbody>
</table>

SEE ALSO

ufsdump(1M), ufsrestore(1M), ctime(3C), printf(3C), scanf(3C), attributes(5), types(3HEAD)

SEE ALSO

550  man pages section 4: File Formats  •  Last Revised 7 Jan 1994
DESCRIPTION
The file /var/yp/updaters is a makefile (see make(1S)) which is used for updating the Network Information Service (NIS) databases. Databases can only be updated in a secure network, that is, one that has a publickey(4) database. Each entry in the file is a make target for a particular NIS database. For example, if there is an NIS database named passwd.byname that can be updated, there should be a make target named passwd.byname in the updaters file with the command to update the file.

The information necessary to make the update is passed to the update command through standard input. The information passed is described below (all items are followed by a NEWLINE except for 4 and 6):

1. Network name of client wishing to make the update (a string).
2. Kind of update (an integer).
3. Number of bytes in key (an integer).
4. Actual bytes of key.
5. Number of bytes in data (an integer).
6. Actual bytes of data.

After receiving this information through standard input, the command to update the particular database determines whether the user is allowed to make the change. If not, it exits with the status YPERR_ACCESS. If the user is allowed to make the change, the command makes the change and exits with a status of zero. If there are any errors that may prevent the updaters from making the change, it should exit with the status that matches a valid NIS error code described in <rpcsvc/ypclnt.h>.

FILES
/var/yp/updaters  The makefile used for updating the NIS databases.

SEE ALSO
make(1S), rpc.ypupdated(1M), publickey(4)

NOTES
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. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.
user_attr(4)

NAME  user_attr – extended user attributes database

SYNOPSIS  /etc/user_attr

DESCRIPTION  /etc/user_attr is a local source of extended attributes associated with users and roles. user_attr can be used with other user attribute sources, including the user_attr NIS map and NIS+ table. Programs use the getuserattr(3SECDB) routines to gain access to this information.

The search order for multiple user_attr sources is specified in the /etc/nsswitch.conf file, as described in the nsswitch.conf(4) man page. The search order follows that for passwd(4).

Each entry in the user_attr databases consists of a single line with five fields separated by colons (:). Line continuations using the backslash (\) character are permitted. Each entry has the form:

```
user:qualifier:res1:res2:attr
```

user  The name of the user as specified in the passwd(4) database.

qualifier  Reserved for future use.

res1  Reserved for future use.

res2  Reserved for future use.

attr  An optional list of semicolon-separated (;) key-value pairs that describe the security attributes to apply to the object upon execution. Zero or more keys may be specified. There are five valid keys: auths, profiles, roles, type, and project.

auths  Specifies a comma-separated list of authorization names chosen from those names defined in the auth_attr(4) database. Authorization names may be specified using the asterisk (*) character as a wildcard. For example, solaris.printer.* means all of Sun’s printer authorizations.

profiles  Contains an ordered, comma-separated list of profile names chosen from prof_attr(4). Profiles are enforced by the profile shells, pfcsh, pfksh, and pfsh. See pfsh(1). If no profiles are assigned, the profile shells do not allow the user to execute any commands.

roles  Can be assigned a comma-separated list of role names from the set of user accounts in this database whose type field indicates the
account is a role. If the roles key value is not specified, the user is not permitted to assume any role.

**type**

Can be assigned one of these strings: normal, indicating that this account is for a normal user, one who logs in; or role, indicating that this account is for a role. Roles can only be assumed by a normal user after the user has logged in.

**project**

Can be assigned a name of one project from the project(4) database to be used as a default project to place the user in at login time. For more information, see getdefaultproj(3EXACCT).

**EXAMPLE 1 Assigning a Profile to Root**

The following example entry assigns to root the All profile, which allows root to use all commands in the system, and also assigns two authorizations:

```
root:::auths=solaris.*,solaris.grant;profiles=All;type=normal
```

The solaris.* wildcard authorization shown above gives root all the solaris authorizations; and the solaris.grant authorization gives root the right to grant to others any solaris authorizations that root has. The combination of authorizations enables root to grant to others all the solaris authorizations. See auth_attr(4) for more about authorizations.

**FILES**

/etc/nsswitch.conf

/etc/user_attr

**NOTES**

When deciding which authorization source to use, keep in mind that NIS+ provides stronger authentication than NIS.

The root user is usually defined in local databases for a number of reasons, including the fact that root needs to be able to log in and do system maintenance in single-user mode, before the network name service databases are available. For this reason, an entry should exist for root in the local user_attr file, and the precedence shown in the example nsswitch.conf(4) file entry under EXAMPLES is highly recommended.

Because the list of legal keys is likely to expand, any code that parses this database must be written to ignore unknown key-value pairs without error. When any new keywords are created, the names should be prefixed with a unique string, such as the company’s stock symbol, to avoid potential naming conflicts.

In the attr field, escape the following symbols with a backslash (\) if you use them in any value: colon (:), semicolon (;), carriage return (\n), equals (=), or backslash (\).
user_attr(4)

SEE ALSO auths(1), pfcsh(1), pfksh(1), pfsh(1), profiles(1), roles(1),
getdefaultproj(3EXACCT), getuserattr(3SECDB), auth_attr(4),
exec_attr(4), nsswitch.conf(4), passwd(4), prof_attr(4), project(4)
utmp(4)

<table>
<thead>
<tr>
<th>NAME</th>
<th>utmp, wtmp – utmp and wtmp database entry formats</th>
</tr>
</thead>
</table>
| SYNOPSIS      | `#include <utmp.h>`  
                | `/var/adm/utmp`  
                | `/var/adm/wtmp` |
| DESCRIPTION   | The utmp and wtmp database files are obsolete and are no longer present on the system. They have been superseded by the extended database contained in the utmpx and wtmpx database files. See utmpx(4). It is possible for /var/adm/utmp to reappear on the system. This would most likely occur if a third party application that still uses utmp recreates the file if it finds it missing. This file should not be allowed to remain on the system. The user should investigate to determine which application is recreating this file. |
| SEE ALSO      | utmpx(4) |

File Formats  555
**NAME**

utmpx, wttmpx – utmpx and wttmpx database entry formats

**SYNOPSIS**

```
#include <utmpx.h>
/var/adm/utmpx
/var/adm/wtmpx
```

**DESCRIPTION**

The `utmpx` and `wtmpx` files are extended database files that have superseded the obsolete `utmp` and `wttmp` database files.

The `utmpx` database contains user access and accounting information for commands such as `who(1)`, `write(1)`, and `login(1)`. The `wtmpx` database contains the history of user access and accounting information for the `utmpx` database.

**USAGE**

Applications should not access these databases directly, but should use the functions described on the `getutxent(3C)` manual page to interact with the `utmpx` and `wtmpx` databases to ensure that they are maintained consistently.

**FILES**

```
/var/adm/utmpx user access and administration information
/var/adm/wtmpx history of user access and administrative information
```

**SEE ALSO**

`wait(2)`, `getutxent(3C)`, `wstat(3XFN)`
NAME
vfstab – table of file system defaults

DESCRIPTION
The file /etc/vfstab describes defaults for each file system. The information is stored in a table with the following column headings:

```
device device mount FS fsck mount mount
to mount to fsck point type pass at boot options
```

The fields in the table are space-separated and show the resource name (device to mount), the raw device to fsck (device to fsck), the default mount directory (mount point), the name of the file system type (FS type), the number used by fsck to decide whether to check the file system automatically (fsck pass), whether the file system should be mounted automatically by mountall (mount at boot), and the file system mount options (mount options). (See respective mount file system man page below in SEE ALSO for mount options.) A '-' is used to indicate no entry in a field. This may be used when a field does not apply to the resource being mounted.

The getvfsent(3C) family of routines is used to read and write to /etc/vfstab.

/etc/vfstab may be used to specify swap areas. An entry so specified, (which can be a file or a device), will automatically be added as a swap area by the /sbin/swapadd script when the system boots. To specify a swap area, the device-to-mount field contains the name of the swap file or device, the FS-type is "swap", mount-at-boot is "no" and all other fields have no entry.

SEE ALSO
fsck(1M), mount(1M), mount_cachefs(1M), mount_hsfs(1M), mount_nfs(1M), mount_tmpfs(1M), mount_ufs(1M), swap(1M), getvfsent(3C)

File Administration Guide, Volume 1
vold.conf(4)

NAME  vold.conf – Volume Management configuration file

SYNOPSIS  /etc/vold.conf

DESCRIPTION  The vold.conf file contains the Volume Management configuration information used by vold(1M). This information includes the database to use, labels that are supported, devices to use, actions to take when certain media events occur, and the list of file systems that are unsafe to eject without unmounting.

Modify vold.conf to specify which program should be called when media events happen (actions) or when you need to add another device to your system. See the example section for more information on adding devices.

If you modify vold.conf, you must tell vold to reread vold.conf by sending a HUP signal. Use

# ps -ef | grep vold
# kill -HUP vold_pid

File Format  The syntax for the vold.conf file is shown here.

# Database to use
db database

# Labels supported
label label_type shared_object device

# Devices to use
use device type special shared_object synname [ options ]

# Actions
insert regex [ options ] program program args
eject regex [ options ] program program args
notify regex [ options ] program program args

# List of file system types unsafe to eject
unsafe fs_type fs_type

Of these syntax fields, you can safely modify Devices to use and Actions.

Devices to Use Field  All use device statements must be grouped together by device type. (For example, all use cdrom statements must be grouped together; and all use floppy statements must be grouped together.) Here are the explanations of the syntax for the Devices to use field.

device  The type of removable media device to be used. Legal values are cdrom, floppy, pcmem and rmdisk.

type  The specific capabilities of the device. Legal value is drive.
This \texttt{sh}(1) expression specifies the device or devices to be used. Path usually begins with \texttt{/dev}.

The name of the program that manages this device. \texttt{vold(1M)} expects to find this program in \texttt{/usr/lib/vold}.

The symbolic name that refers to this device. The \texttt{symname} is placed in the device directory.

The user, group, and mode permissions for the media inserted (optional).

The \texttt{special} and \texttt{symname} parameters are related. If \texttt{special} contains any shell wildcard characters (i.e., has one or more asterisks or question marks in it), then the \texttt{symname} must have a \texttt{"%d"} at its end. In this case, the devices that are found to match the regular expression are sorted, then numbered. The first device will have a zero filled in for the \texttt{"%d"}, the second device found will have a one, and so on.

If the \texttt{special} specification does not have any shell wildcard characters then the \texttt{symname} parameter must explicitly specify a number at its end (see \texttt{EXAMPLES} below).

Here are the explanations of the syntax for the \texttt{Actions} field.

- \texttt{insert}\mid\texttt{eject}\mid\texttt{notify}: The media event prompting the event
- \texttt{regex}: This \texttt{sh}(1) regular expression is matched against each entry in the \texttt{/vol} file system that is being affected by this event.
- \texttt{options}: You can specify what user or group name that this event is to run as (optional).
- \texttt{program}: The full path name of an executable program to be run when \texttt{regex} is matched.
- \texttt{program args}: Arguments to the program.

The default \texttt{vold.conf} file is shown here.

```bash
# # Volume Daemon Configuration file # # Database to use (must be first) db db_mem.so # Labels supported label dos label_dos.so floppy label cdom label_cdom.so cdom label sun label_sun.so floppy # Devices to use use cdom drive /dev/dsk/c*s2 dev_cdom.so cdom%d use floppy drive /dev/diskette[0-9] dev_floppy.so floppy%d
```

\textbf{Actions Field}

\textbf{Default Values}
vold.conf(4)

```
# Actions
insert /vol*/dev/fd[0-9]/* user=root /usr/sbin/rmmount
insert /vol*/dev/dsk/* user=root /usr/sbin/rmmount
eject /vol*/dev/fd[0-9]/* user=root /usr/sbin/rmmount
eject /vol*/dev/dsk/* user=root /usr/sbin/rmmount
notify /vol*/rdsk/* group/tty user=root /usr/lib/vold/volmissing -p

# List of file system types unsafe to eject
unsafe ufs hsfs pcfs
```

**EXAMPLE 1**

A sample `vold.conf` file.

To add a CD-ROM drive to the `vold.conf` file that does not match the default regular expression `/dev/rdsk/c*s2`, you must explicitly list its device path and what symbolic name (with `%d`) you want the device path to have. For example, to add a CD-ROM drive that has the path `/dev/rdsk/my/cdrom` (where `s?` are the different slices), add the following line to `vold.conf` (all on one line):

```
use cdrom drive /dev/rdsk/my/cdroms2 dev_cdrom.so cdrom%d
```

Then, when a volume is inserted in this CD-ROM drive, volume management will assign it the next symbolic name. For example, if two CD-ROMs match the default regular expression, they would be named `cdrom0` and `cdrom1`; and any that match the added regular expression would be named starting with `cdrom2`.

For a diskette that does not match the `vold.conf` default regular expression `/dev/floppy[0-9]`, a similar line would have to be added for the diskette. For example, to add a diskette whose path was `/dev/my/fd0`, you would add the following to `vold.conf`:

```
use floppy drive /dev/my/fd0 dev_floppy.so floppy%d
```

**SEE ALSO**

`sh(1), volcancel(1), volcheck(1), volmissing(1), rmmount(1M), vold(1M), rmmount.conf(4), volfs(7FS)`

**NOTES**

Volume Management manages both the block and character device for CD-ROMs and floppy disks; but, to make the configuration file easier to set up and scan, only one of these devices needs to be specified. If you follow the conventions specified below, Volume Management figures out both device names if only one of them is specified. For example, if you specify the block device, it figures out the pathname to the character device; if you specify the pathname to the character device, it figures out the block device.

**CD-ROM Naming Conventions**

The CD-ROM pathname must have a directory component of `rdsk` (for the character device) and `dsk` for the block device. For example, if you specify the character device using the line:

```
use cdrom drive /dev/rdsk/my/cdroms2 dev_cdrom.so cdrom%d
```
then it is assumed that the block device is at

/dev/dsk/my/cdroms2

For floppy disks, Volume Management requires that the device pathnames end in either rfd[0-9] or rdiskette[0-9] for the character device, and fd[0-9] or diskette[0-9] for the block device. As with the other removable disks, it generates either the block name given the character name, or the character name given the block name.
NAME
warn.conf – Kerberos warning configuration file

SYNOPSIS
/etc/krb5/warn.conf

DESCRIPTION
The warn.conf file contains configuration information specifying how users will be warned by the ktkt_warnd daemon about ticket expiration on a Kerberos client. Each Kerberos client host must have a warn.conf file in order for users on that host to get Kerberos warnings from the client. Entries in the warn.conf file must have the following format:

principal syslog | terminal | mail time [email_address]

principal
The principal name to be warned. The '*' wildcard can be used to specify groups of principals.
syslog
Sends the warnings to the system’s syslog. Depending on the /etc/syslog.conf file, syslog entries are written to the /var/adm/messages file and/or displayed on the terminal.
terminal
Sends the warnings to display on the terminal.
mail
Sends the warnings as email to the address specified by email_address.
time
Specifies how much time before the TGT expires when a warning should be sent. The default time value is seconds, but you can specify h (hours) and m (minutes) after the number to specify other time values.
email_address
Specifies the email address at which to send the warnings. This field must be specified only with the mail field.

EXAMPLES
EXAMPLE 1 Specifying warnings

The following warn.conf entry specifies that warnings will be sent to the syslog 5 minutes before the expiration of the TGT for all principals, in the form:
"jdb@ACME.COM: your kerberos credentials expire in 5 minutes".

* syslog 5m

FILES
/usr/lib/krb5/ktkt_warnd Kerberos warning daemon

SEE ALSO
ktkt_warnd(1M), SEAM(5)
ypfiles – Network Information Service Version 2, formerly known as YP

**DESCRIPTION**

The NIS network information service uses a distributed, replicated database of dbm files (in ASCII form) contained in the `/var/yp` directory hierarchy on each NIS server. NIS has been replaced by NIS+, the new version of the Network Information Service. See `nis+(1)`. This release only supports the client functionality of NIS, (see `ypclnt(3NSL)`). The client functions are either supported by the `ypserv` process running on a machine with an earlier version of SunOS or by the NIS+ server in "YP-compatibility" mode, (see `rpc.nisd(1M)`).

A dbm database served by the NIS server is called an NIS map. An NIS domain is a subdirectory of `/var/yp` containing a set of NIS maps on each NIS server.

Standard nicknames are defined in the file `/var/yp/nicknames`. These names can be used in place of the full map name in the `ypmatch` and `ypcat` commands. The command `ypcat -x` can be used to display the current set of nicknames. The command `ypwhich -m` can be used to display all the available maps. Each line of the nickname file contains two fields separated by white space. The first field is the nickname and the second field is the name of the map that it expands to. The nickname cannot contain a ".".

**FILES**

`/var/yp/nicknames` nicknames file

**SEE ALSO**

`nis+(1), nisaddent(1M), nissetup(1M), rpc.nisd(1M), ypbind(1M), ypinit(1M), dbm(3UCB), secure_rpc(3NSL), ypclnt(3NSL)`

**NOTES**

The NIS+ server, `rpc.nisd`, when run in "YP-compatibility mode", can support NIS clients only for the standard NIS maps listed below, provided that it has been set up to serve the corresponding NIS+ tables using `nissetup(1M)` and `nisaddent(1M)`. The NIS+ server should serve the directory with the same name (case sensitive) as the domainname of the NIS client. NIS+ servers use secure RPC to verify client credentials but the NIS clients do not authenticate their requests using secure RPC. Therefore, NIS clients can look up the information stored by the NIS+ server only if the information has "read" access for an unauthenticated client (i.e. one with "nobody" NIS+ credentials).

**NIS maps**

<table>
<thead>
<tr>
<th>NIS maps</th>
<th>NIS+ tables</th>
</tr>
</thead>
<tbody>
<tr>
<td>passwd.byname</td>
<td>passwd.org_dir</td>
</tr>
<tr>
<td>passwd.byuid</td>
<td>passwd.org_dir</td>
</tr>
<tr>
<td>group.byname</td>
<td>group.org_dir</td>
</tr>
<tr>
<td>group.bygid</td>
<td>group.org_dir</td>
</tr>
<tr>
<td>publickey.byname</td>
<td>cred.org_dir</td>
</tr>
<tr>
<td>hosts.byaddr</td>
<td>hosts.org_dir</td>
</tr>
<tr>
<td>hosts.byname</td>
<td>hosts.org_dir</td>
</tr>
<tr>
<td>mail.byaddr</td>
<td>mail_aliases.org_dir</td>
</tr>
<tr>
<td>File</td>
<td>Directory</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>mail.aliases</td>
<td>mail_aliases.org_dir</td>
</tr>
<tr>
<td>services.byname</td>
<td>services.org_dir</td>
</tr>
<tr>
<td>services.byservicename</td>
<td>services.org_dir</td>
</tr>
<tr>
<td>rpc.bynumber</td>
<td>rpc.org_dir</td>
</tr>
<tr>
<td>rpcbyname</td>
<td>rpc.org_dir</td>
</tr>
<tr>
<td>protocols.bynumber</td>
<td>protocols.org_dir</td>
</tr>
<tr>
<td>protocolsbyname</td>
<td>protocols.org_dir</td>
</tr>
<tr>
<td>networks.byaddr</td>
<td>networks.org_dir</td>
</tr>
<tr>
<td>networksbyname</td>
<td>networks.org_dir</td>
</tr>
<tr>
<td>netmasks.bymask</td>
<td>netmasks.org_dir</td>
</tr>
<tr>
<td>netmasks.byaddr</td>
<td>netmasks.org_dir</td>
</tr>
<tr>
<td>ethersbyname</td>
<td>ethers.org_dir</td>
</tr>
<tr>
<td>ethersbyaddr</td>
<td>ethersbyname</td>
</tr>
<tr>
<td>bootparams</td>
<td>bootparams</td>
</tr>
<tr>
<td>auto.master</td>
<td>auto_master.org_dir</td>
</tr>
<tr>
<td>auto.home</td>
<td>auto_home.org_dir</td>
</tr>
<tr>
<td>auto.direct</td>
<td>auto_direct.org_dir</td>
</tr>
<tr>
<td>autosrc</td>
<td>autosrc.org_dir</td>
</tr>
<tr>
<td>NAME</td>
<td>zoneinfo – timezone information</td>
</tr>
<tr>
<td>----------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>For notes regarding the zoneinfo timezones, see /usr/share/lib/zoneinfo/src/README.</td>
</tr>
</tbody>
</table>
zoneinfo(4)
Index

C
configuration file for pluggable authentication modules — pam.conf, 300

A
a.out — Executable and Linking (ELF) files, 26
accounting system, prime/nonprime hours —
holidays, 154
addresses — addresses for sendmail, 22
admin — installation defaults file, 18
alias — alias table file of encoding names, 21
alias table file of encoding names — alias, 21
aliases — sendmail alias file, 22
archives — device header, 28
ASET environment file — asetenv, 31
ASET master files
— asetmasters, 33
— cklst.high, 33
— cklst.low, 33
— cklst.med, 33
— tune.high, 33
— tune.low, 33
— tune.med, 33
— uid_aliases, 33
asetenv — ASET environment file, 31
audit_class password file, 36
audit_event password file, 42
audit — audit control file, 38
audit — audit data file, 41
audit.log — audit trail file, 43
audit trail file, — audit.log, 43
audit_user — per-user auditing data file, 49

B
boot parameter database — bootparams, 54
BOOTP, network database —
dhcp.network, 91
bootparams — boot parameter database, 54

C
CD-ROM table of contents file — cdtoc, 57
cdtoc — CD-ROM table of contents file, 57
.clustertoc — listing of software packages on
product distribution media, 60
compatible versions file — compver, 64
compver — compatible versions file, 64
configuration file, system log daemon —
syslogd, 461
configuration file for default router(s) —
defaultrouter, 72
configuration file for in.named —
named.conf, 220
configuration file for LDAP display template
routines, — ldaptemplates.conf, 192
configuration file for LDAP filtering routines, —
ldapfilter.conf, 187
configuration file for LDAP search preference
routines, — ldapsearchprefs.conf, 189
configuration file for Mobile IP mobility agent
— mipagent.conf, 212
Solaris Network Cache and Accelerator (NCA)
socket utility library — nca_addr, 248
configuration file for NIS security —
curenets, 433
configuration file for security policy —
policy.conf, 333
configuration file for Service Location Protocol
agents — slp.conf, 440
copyright — copyright information file, 65
core — core image of a terminated process
file, 66

d_d_passwd — dial-up password file, 103
Generating An Encrypted Password, 104
default_fs — specify the default file system type
for local or remote file systems, 71
default Internet protocol type — inet_type, 163
defaultrouter — configuration file for default
router(s), 72
depend — software dependencies file, 73
devconfig configuration files —
device.cinfo, 77
device_allocate, device access control file, 75
device.cinfo — devconfig configuration
files, 77
device instance number file —
path_to_inst, 310
device_maps, device access control file, 82
devices
access control file — device_allocate, 75
access control file — device_maps, 82
devices, capabilities, terminal and printers —
termino, 472
dfs utilities packages, list — fstypes, 128
dfstab — file containing commands for sharing
resources, 84
DHCP
client identifier to IP address mappings —
dhcip_network, 91
configuration parameter table — dhcptab, 97
dhcip_network — DHCP network database, 91
dhcip_network
See also pntadm
dhcpsvc.conf — file containing service
configuration parameters for the DHCP
service, 94
dhcptab — DHCP configuration parameter
table, 97
dial-up password file — d_passwd, 103
dialups — list of terminal devices requiring a
dial-up password, 101
dir_ufs — format of ufs directories, 102
directory of files specifying supported platforms
— platform, 327
disk drive configuration for the format
command — format.dat, 122
disk space requirement file — space, 451
dispatcher, real-time process, parameters —
rt_dptbl, 420
dispatcher, time-sharing process, parameters —
ts_dptbl, 535
driver.conf — driver configuration file, 105
drivers
driver for EISA devices — eisa, 453
driver for ISA devices — isa, 453
driver for PCI devices — pci, 312
driver for pseudo devices — pseudo, 391
driver for SBus devices — vme, 425
driver for SCSI devices — scsi, 431

eisa — configuration file for EISA bus device
drivers, 453
ELF files — a.out, 26
.environ — user-preference variables files for
AT&T FACE, 108
environ — user-preference variables files for
AT&T FACE, 108
evironment, setting up an environment for
user at login time — profile, 381
ethers — Ethernet addresses of hosts on
Internet, 110
exec.attr — execution profiles database, 111
Executable and Linking Format (ELF) files —
a.out, 26
execution profiles database — exec.attr, 111
FACE
  alias file — pathalias, 309
  object architecture information — ott, 292
FACE object architecture information,— ott, 292
fd — file descriptor files, 113
file containing service configuration parameters for the DHCP service — dhcpsvc.conf, 94
file descriptor files — fd, 113
file formats — intro, 16
file listing users to be disallowed ftp login privileges — ftpusers, 132
file system
defaults — vfstab, 557
mounted— mtab, 217
file that maps sockets to transport providers — sock2path, 450
files used by programs
/etc/security/device_allocate — device_allocate file, 76
/etc/security/device_maps — device_maps file, 82
flash_archive — format of flash archive, 114
format of a ufs file system volume — fs_ufs, 129
inode, 129
inode_ufs, 129
format.dat — disk drive configuration for the format command, 122
Keywords, 122
Syntax, 122
format of flash archive — flash_archive, 114
forward — mail forwarding file, 22
fs_ufs — format of a ufs file system volume, 129
fspec — format specification in text files, 126
fstypes — file that lists utilities packages for distributed file system, 128
ftpusers — file listing users to be disallowed ftp login privileges, 132
geniconvtbl — geniconvtbl input file format — geniconvtbl, 134
geniconvtbl input file format — geniconvtbl, 134
graphics interface files — plot, 331
group — local source of group information, 152
holidays — prime/nonprime hours for accounting system, 154
host name database — hosts, 155
hosts.equiv — trusted hosts list, 157
hosts — host name data base, 155
inet_type — default Internet protocol type, 163
inetcdf.conf — Internet server database, 160
init.d — initialization and termination scripts for changing init states, 164
initialization and termination scripts for changing init states — init.d, 164
inittab — script for init, 166
inode — format of a ufs file system volume, 129
inode_ufs — format of a ufs file system volume, 129
installation, defaults file — admin, 18
Internet
dhcp_network, 91
DHCP database — dhcp_network, 91
Ethernet addresses of hosts — ethers, 110
network name database — networks, 271
protocol name database — protocols, 384
services and aliases — services, 435
Internet servers database — servers, 160
ipnodes — local database associating names of nodes with IP addresses, 169
isa — configuration file for ISA bus device drivers, 453
issue — issue identification file, 171
Kerberos configuration file — krb5.conf, 179
Kerberos warning configuration file — warn.conf, 562
keyboard table descriptions for loadkeys and dumpkeys — keytables, 172
keytables — keyboard table descriptions for loadkeys and dumpkeys, 172
krb5.conf — Kerberos configuration file, 179

L
ldapfilter.conf — configuration file for LDAP filtering routines, 187
ldapsearchprefs.conf — configuration file for LDAP search preference routines, 189
ldap_templates.conf — configuration file for LDAP display template routines, 192
legal annotations, specify — note, 280
limits — header for implementation-specific constants, 196
link editor output — a.out, 26
list of boot environments — lutab, 208
list of network groups — netgroup, 264
list of terminal devices requiring a dial-up password — dialups, 101
llc2 — LLC2 Configuration file, 200
LLC2 Configuration file — llc2, 200
local database associating names of nodes with IP addresses — ipnodes, 169
login-based device permissions — logindevperm, 206
logindevperm — login-based device permissions, 206
loginlog — log of failed login attempts, 207
lutab — list of boot environments, 208

M
magic — file command’s magic numbers table, 209
message displayed to users attempting to log on in the process of a system shutdown — nologin, 279
mipagent.conf — configuration file for Mobile IP mobility agent, 212
ncad_addr — Solaris Network Cache and Accelerator (NCA) socket utility library, 248
mounted file system table — mtab, 217
mtab — mounted file system table, 217

N
name service cache daemon configuration — nscd.conf, nscd.conf, 281
name service switch, configuration file — nsswitch.conf, 283
named.conf — configuration file for in.named, 220
cac.conf — the NCA configuration file that specifies physical interfaces, 249
ncakmod.conf — the ncakmod configuration file, 251
ncalogd.conf — the ncalogd configuration file, 253
netconfig — network configuration database, 259
netgroup — list of network groups, 264
netid — netname database, 266
netmasks — network masks for subnetting, 268
netname database — netid, 266
.netrc — ftp remote login data file, 270
Network Information Service Version 2, formerly known as YP — ypfiles, 563
networks connected to the system — netconfig, 259
networks — network name database, 271
NFS, remote mounted file systems — rmtab, 416
NIS databases, updating — updaters, 551
nisfiles — NIS+ database files and directory structure, 275
nologin — message displayed to users attempting to log on in the process of a system shutdown, 279
nonprime hours, accounting system — holidays, 154
note — specify legal annotations, 280
nscd.conf — name service cache daemon configuration, 281
nsswitch.conf — configuration file for the name service switch, 283

O
.order — installation order of software packages on product distribution media, 291
package characteristics file, — pkginfo, 318
package contents description file, —
   pkgmap, 324
package information file — prototype, 386
package installation order file, — order, 291
package table of contents description file
   .clustertoc — clustertoc, 60
   — package toc, 293
   .package toc — listing of software packages on
      product distribution media, 293
   packing rules file for cachefs and filesync —
      packingrules, 297
   packingrules — packing rules file for cachefs
      and filesync, 297
   pam.conf — configuration file for pluggable
      authentication modules, 300
   passwd — password file, 306
   passwords, access-restricted shadow system file
      — shadow, 436
   path_to_inst — device instance number
      file, 310
   pathalias — alias file for FACE, 309
   PCI devices, driver class — pci, 312
   pci — drivers for PCI devices, 312
   pcmcia — PCMCIA nexus driver, 316
   PCMCIA nexus driver — pcmcia, 316
   per-user auditing data file — audit_user, 49
   phones — remote host phone numbers, 317
   pkginfo — software package characteristics
      file, 318
   pkgmap — listing of software package contents, 324
   platform — directory of files specifying
      supported platforms, 327
   plot — graphics interface files, 331
   policy.conf — configuration file for security
      policy, 333
   power.conf — Power Management
      configuration file, 334
   Power Management configuration file —
      power.conf, 334
   .pref — user-preference variables files for AT&T
      FACE, 108
   prime hours, accounting system —
      holidays, 154
   printers.conf — printing configuration
      database, 345
   printers — printer alias database, 342
   proc — /proc, the process file system, 351
   proc — process file system, 351
   proc — /proc, the process file system
      PCAGENT, 373
      PCCFAULT, 369
      PCCSIG, 367
      PCKILL, 368
      PCNICE, 374
      PCREAD PCWRITE, 373
      PCRUN, 366
      PCSASRS, 373
      PCSCRED, 374
      PCSENTRY PCSEEXIT, 369
      PCSET PCUNSET, 371
      PCFAULT, 368
      PCSFPREG, 372
      PCHOLD, 368
      PCSREG, 372
      PCSSIG, 372
      PCSTOP PCWSTOP
       PCWSTOP, 366
       PCSTRACE, 367
       PCSVADDR, 372
       PCSXREG, 373
       PCUNKILL, 368
       PCWATCH, 369
       /proc, the process file system — proc, 351
   process file system — proc, 351
   process scheduler (or dispatcher), real-time,
      parameters — rt_dptbl, 420
   process scheduler (or dispatcher), time-sharing,
      parameters — ts_dptbl, 535
   processes, core image of a terminated process
      file — core, 66
   profile — setting up an environment for user at
      login time, 381
   project — project file, 382
   project file — project, 382
   project identification file — issue, 171
   protocols — names of known protocols in
      Internet, 384
   prototype — package information file, 386
   pseudo devices, 391
   pseudo — drivers for pseudo devices, 391
   publickey — publickey database for secure
      RPC, 392
   Index  571
Q
queuedefs — queue description file for at, batch, and cron spooled by at or batch or atrm, 393

R
real-time process dispatcher, parameters — rt_dptbl, 420
real-time process scheduler, parameters — rt_dptbl, 420
remote authentication for hosts and users — hosts.equiv, .rhosts, 157
remote — remote host descriptions, 405
remote host, phone numbers — phones, 317
remote login data for ftp — netrc, 270
remote mounted file systems — rmtab, 416
Remote Program Load (RPL) server configuration file — rpld.conf, 418
resolv.conf — resolver configuration file — resolv.conf, 409
rmmount.conf — removable media mounter configuration file
  Default Values, 412
Examples, 412
rpc — rpc program number database, 417
RPC program names, for program numbers — rpc, 417
RPC security, public key database — publickey, 392
RPCSEC_GSS mechanism file, — mech, 211
RPCSEC_GSS QOP file, —, 211
rpld.conf — Remote Program Load (RPL) server configuration file, 418

S
SBus devices, driver class — sbus, 425
sbus — drivers for SBus devices, 425
sccsfile — format of SCCS history file, 428
scheduler, real-time process, parameters — rt_dptbl, 420
scheduler, time-sharing process, parameters — ts_dptbl, 535
SCSI devices, driver class — ssci, 431
ssci — drivers for SCSI devices, 431
securenets — configuration file for NIS security, 433
sendmail addresses file — addresses, 22
sendmail aliases file — aliases, 22
sendmail aliases file — forward, 22
serialized registration file for the service location protocol daemon (slpd) —
slpd.reg, 448
services — Internet services and aliases, 435
shadow password file, 436
share resources across network, commands — dfstab, 84
shared resources, local, — sharetab, 438
sharetab — shared file system table, 438
shell database — shells, 439
shells — shell database, 439
slp.conf — configuration file for Service Location Protocol agents, 440
slpd.reg — serialized registration file for the service location protocol daemon (slpd), 448
sock2path — file that maps sockets to transport providers, 450
software dependencies — depend, 73
space — disk space requirement file, 451
specify the default file system type for local or remote file systems — default_fs, 71
su command log file — sulog, 452
sulog — su command log file, 452
sysbus — configuration files for ISA and EISA bus device drivers, 453
sysbus — drivers for system bus, 453
sysbus — drivers for EISA devices, eisa, 453
sysbus — drivers for ISA devices, isa, 453
sysidcfg — system identification configuration file, 456
  Keyword Syntax Rules, 456
  Where To Put the sysidcfg File, 456
syslogd.conf — system log daemon configuration file, 461
system — system configuration information, 464
system identification configuration file — sysidcfg, 456
system log configuration file — syslogd.conf, 461

572  man pages section 4: File Formats • February 2004
telnet default options file — telnetrc, 468
telnetrc — file for telnet default options, 468
term — format of compiled term file, 469
terminals, line setting information —
ttydefs, 542
termination and initialization scripts for changing init states — init.d, 164
terminfo — System V terminal capability data base, 472
test files, format specification — fspec, 126
the NCA configuration file that specifies physical interfaces — nca.if, 249
the ncakmod configuration file —
ncakmod.conf, 251
the ncalogd configuration file —
ncalogd.conf, 253
time-sharing process dispatcher, parameters —
ts_dptbl, 535
time-sharing process scheduler, parameters —
ts_dptbl, 535
timezone — set default time zone, 526
timed event services, queue description file for at, batch and cron — queuedefs, 393
timezone — default timezone data base, 527
timezone information — zoneinfo, 565
TNF kernel probes — tnf_kernel_probes, 528
tnf_kernel_probes — TNF kernel probes, 528
ttydefs — terminal line settings information, 542
ttyname, list of directories with terminal-related device files — ttysrch, 543

ufs, format — dir_ufs, 102
ufs dump — incremental dump format, 545
updaters — configuration file for NIS updating, 551
user-preference variables files for AT&T FACE — environ, 108
utmp — utmp and wtmp database entry formats, 555
utmp and wtmp database entry formats —
utmp, 555
utmp and wtmp database entry formats —
wtmp, 556

warn.conf — Kerberos warning configuration file, 562
wtmp — utmp and wtmp database entry formats, 555
wtmpx — utmpx and wtmpx database entry formats, 556
wtmp and wtmpx database entry formats —
wtmpx, 556

zoneinfo — timezone information, 565